ARMY RESEARCH LABORATORY

Computer Programs for LB/TS Test Design: Technical Description, Usage Instructions and Source Code Listings

Stephen J. Schraml
Richard J. Pearson

ARL-MR-260 September 1995

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

NOTICES

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Final, Oct 94 - May 95

4. TITLE AND SUBTITLE
Computer Programs for LB/TS Test Design: Technical Description, Usage Instructions and Source Code Listings

5. FUNDING NUMBERS
4G017-501-U2

6. AUTHOR(S)
Stephen J. Schraml and Richard J. Pearson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Research Laboratory
ATTN: AMSRL-WT-NC
Aberdeen Proving Ground, MD 21005-5066

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-MR-260

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Computer programs developed to support the design and operation of the Large Blast/Thermal Simulator (LB/TS) are presented. A technical description of each program is included in the discussion. Examples of program usage and source code listings are provided in the appendices. The purpose of this report is to serve as a vehicle to transfer LB/TS test design and analysis technology to the Defense Nuclear Agency and its contractors.

14. SUBJECT TERMS
blast, shock tubes, nuclear weapons, computer programming

15. NUMBER OF PAGES
91

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Intentionally Left Blank

Table of Contents

Intentionally Left Blank

1. Introduction

The Large Blast/Thermal Simulator (LB/TS) is a facility for testing military equipment in the blast and thermal radiation environment produced by tactical nuclear weapons. The facility is located at the White Sands Missile Range, New Mexico. The LB/TS is the world's largest shock tube and is specially configured and operated to produce high fidelity simulations of ideal[1] nuclear blast waveforms.[2]

Beginning with the initial concept and through the design phase of the LB/TS, it was necessary to develop software tools that would be used in estimating the facility's size and performance requirements for meeting the anticipated blast testing objectives. After construction, additional utilities were needed to support facility characterization and regular test operation.

This report describes several of the utility programs that have been developed at the U.S. Army Research Laboratory (ARL) and its predecessor organization, the Ballistic Research Laboratory (BRL). The purpose of this document is to transition use of these tools to the Defense Nuclear Agency (DNA), the U.S. Army Test and Evaluation Command (TECOM) and their contractors who support the LB/TS.

Source listings of the computer programs described in this report are provided in the appendices, along with example calculations. To obtain a diskette of these programs, contact the authors at


```
Director
U.S. Army Research Laboratory
ATTN: AMSRL-WT-NC, Mr. Stephen J. Schraml
Aberdeen Proving Ground, MD 21005-5066
```


2. Ideal Blast Waveforms

Because the LB/TS was designed to simulate ideal nuclear blast waves, it is useful to have a tool that will produce static and dynamic pressure histories associated with the detonation of a weapon over an ideal surface. One such code used by ARL is BLAST, that calculates pressure histories using the modified Freidlander equation and blast wave parameters predicted by the Reflect-4 code that are tabulated for a 40 kT reference blast wave.[3] The form of the modified Freidlander equation used is

$$p(t) = p_{max} * (1 - t/ppd) * e^{-ci*t/ppd} \qquad (1)$$

in which p is the pressure at a particular point in time, t is the time, p_{max} is the amplitude of the incident shock, ppd is the positive phase duration of the blast, and ci is the decay constant of the wave. The resulting blast wave is scaled to the desired weapon yield by Sachs' scaling.[1]

The BLAST program accepts as input the peak static overpressure at the observation point, the yield of the weapon, the ambient pressure, the ambient temperature and the number of records desired in the resulting waveform. Using this input, the program calculates the height of burst (HOB) of the weapon, the ground range from the weapon to the observation point, the arrival time of the shock wave at the observation point, and the static and dynamic pressure histories as a function of time. The code numerically integrates the static and dynamic pressure histories to produce static and dynamic pressure impulse histories, that are also included in the output.

A source code listing of the BLAST program and an example of its use are provided in Appendix A.

3. Ideal Blast Yield

As described above, the BLAST program will produce ideal static and dynamic pressure waveforms for a specified weapon yield. The inverse of this process is to specify the blast conditions recorded at an observation point and determine the yield of the weapon that would produce those conditions in an ideal blast scenario. This is the function of the YIELD program. Like the BLAST program, it uses blast parameters determined by the Reflect-4 code and Sachs' scaling to fit a tabulated reference blast wave to the data specified by the user. The ambient conditions, incident static overpressure, and static and dynamic pressure impulses are provided by the user. Using this information, the code determines the weapon yields required to produce the provided impulses for the given incident overpressure.

A source code listing of the YIELD program and an example of its use are provided in Appendix B.

4. Normal Shock Waves

For cases involving ideal blast in the free field, the blast front can be treated as a normal shock wave in a perfect gas. By assuming that air is a perfect gas with a ratio of specific heats $\gamma = 1.4$, the normal shock wave equations[4] can be employed to determine the properties of the air on either side of the leading shock. The Mach number behind the shock can be determined from

$$M_2^2 = \frac{\gamma + 1}{2\gamma}\frac{p_2}{p_1} + \frac{\gamma - 1}{2\gamma} \tag{2}$$

in which the 2 subscript refers to the shocked air, and the 1 subscript refers to the ambient air. The static density of the shocked air can then be determined from the Rankine-Hugoniot equation

$$\frac{\rho_2}{\rho_1} = \frac{(\gamma + 1)p_2 + (\gamma - 1)p_1}{(\gamma + 1)p_1 + (\gamma - 1)p_2} \tag{3}$$

The stagnation pressure of the shocked air is obtained from the static pressure and Mach number behind the shock from the Pitot equation.

$$\frac{P_2}{p_2} = \left(1 + \frac{\gamma - 1}{2} M_2^2\right)^{\gamma/(\gamma-1)} \tag{4}$$

The SHOCK program, listed in Appendix C, is an interactive Fortran 77 program that prompts the user for the ambient conditions and the strength of the shock (based on pressure or velocity), then computes the shock wave parameters based on the equations above. These parameters are often needed when blast experiments or numerical simulations are being designed.

5. LB/TS Driver Initial Conditions

As previously stated, the LB/TS is a specially configured shock tube. All gas-driven shock tubes consist of a high pressure gas reservoir, called the driver, that is connected to an expansion section. The shock wave is formed by the sudden release of high pressure gas from the driver into the expansion section. The driver system of the LB/TS consists of nine cylindrical drivers that feed into a half-cylinder expansion section. Each driver has an interior diameter of 1.83 m. The volume of each driver can be adjusted, and the maximum available volume of all nine drivers is 583 m^3. The downstream ends of each driver converges to an interior diameter of 0.91 m and end at a double diaphragm system. The expansion section has a diameter of nominally 20 m, with a cross-sectional area of 163 m^2. The expansion section is 170 m in length, with the test section located 101 m from the upstream end of the expansion section.

Flow is initiated by simultaneously rupturing all the diaphragms. The shocks from the nine drivers empty into the expansion section and coalesce into a single, planar shock. The driver gas, which travels at a slower speed than the shock wave, empties into the expansion section and expands to fill the larger cross-sectional area of the tunnel. This expansion causes the density of the gas to increase. The interface between the leading edge of the driver gas and the shocked expansion tunnel gas is referred to as the contact surface. In order to create a high-fidelity, ideal nuclear blast simulation, it is necessary to initially heat the driver gas so that when this expansion takes place, the density is constant across the contact surface.

The positive phase duration (PPD) of a blast wave depends on the time required for the driver gas to empty from the driver system. When long duration blast waves are required, the driver emptying time may be extended by restricting the available flow area of driver gas into the expansion tunnel. The LB/TS has been configured with baffle plates that can be positioned at the diaphragm to perform this task. In addition to extending the PPD of the blast wave, the presence of the baffle plate also reduces the amplitude of the incident shock at the test section.

ARL has employed computational fluid dynamics (CFD) analysis and small scale experimentation to generate empirical relationships between driver gas pressure, temperature

and throat baffle size for proper operation of the LB/TS. Given an appropriate combination of these parameters, the facility will produce a shock wave of a particular amplitude, with density matching across the contact surface between the driver gas and the expansion tunnel gas. These relationships between driver system and blast wave parameters wave been incorporated into a computer program called PTUBE.

The PTUBE program can be operated in a forward mode or a backward mode. When used in the forward mode, the user supplies as input the ambient conditions, the driver pressure, and the baffle size. The program then determines the proper driver temperature to use for matching density across the contact surface, and the expected incident shock overpressure at the test section. In backward mode, the user supplies the ambient conditions, and baffle size, and the desired shock overpressure at the test section and the code determines the proper driver pressure and temperature to create that environment.

A listing of the PTUBE program and examples of its use are provided in Appendix D.

6. LB/TS RWE Function Generation

When the leading shock of the blast wave reaches the downstream end of the expansion tunnel of the LB/TS, a disturbance is created by the abrupt area change from the tunnel to the surrounding atmosphere encountered by the shock front. For shocks with locally subsonic flow behind them, this disturbance causes a rarefaction wave to form at the downstream end of the expansion section that travels upstream, against the direction of the blast flow. The rarefaction wave has a lower pressure on the downstream side than exists on the upstream side and consequently creates an acceleration of the fluid in the expansion section. When this effect reaches the test section, the static pressure decreases and the dynamic pressure increases due to the fluid particle acceleration and the fidelity of the ideal nuclear blast simulation is destroyed.

There are two methods that can be used to eliminate, or delay the arrival of the rarefaction wave at the test section and thereby preserve simulation fidelity. The simplest approach is to increase the length of the expansion section so that the rarefaction wave does not reach the test section until after the period of interest of the simulation is completed. This method is easy to implement on small scale shock tubes, but much too costly on a facility the size of the LB/TS.

The alternative to lengthening the expansion tunnel of the shock tube is to employ a device which will modify the flow in such a way as to make the expansion section appear to be infinitely long. This device is referred to as a rarefaction wave eliminator (RWE) and is an essential component of the LB/TS. To eliminate the formation of a rarefaction wave as the flow passes from the expansion tunnel to the surrounding atmosphere, the fluid is choked by use of an area reduction to accelerate the flow in the reduced area, equalizing the static pressure of the flow to the ambient pressure of the atmosphere. Because the LB/TS produces a decaying blast wave, the conditions that must be satisfied to match the flow pressure to the ambient pressure change continuously with time. A device that performs

this task is referred to as an active RWE because it operates in real time in order to modify the decaying blast wave.

RWE operation is based on the assumption that the flow through it is considered to be the one-dimensional, isentropic flow of an inviscid, ideal gas through a simple, converging nozzle.[5] For isentropic flow through a converging nozzle to reach sonic velocity in expanding from a stagnation pressure of P_{0i} to a static atmospheric pressure of p_∞, the ratio p_∞ / P_{0i} must be larger than the critical ratio given by Equation 5, which is obtained by setting the value of ${M^2}_2$ equal to 1.0 in Equation 4.

$$\frac{p_\infty}{P_{0i}} = \left(\frac{2}{\gamma + 1}\right)^{\frac{\gamma}{\gamma - 1}} \tag{5}$$

If the ratio of specific heats, γ, is equal to 1.400, this critical ratio is 0.528282. If the result of Equation 5 is greater than 0.528282 the flow will remain subsonic at the exit of the RWE converging nozzle. In this case, the rarefaction wave is eliminated by accelerating the fluid such that the static pressure of the flow exiting the RWE will be equal to the ambient pressure. If, on the other hand, the pressure ratio of Equation 5 ratio is less than 0.528282, then with the proper RWE setting the flow will choke and become sonic at the exit of the RWE converging nozzle. In this case, no disturbance will travel upstream into the flow to disrupt the fidelity of the simulation.

These are the two cases to be considered in determining the open area at the exit of the RWE converging nozzle. In one case, the flow remains subsonic and in the other, the flow becomes sonic at the RWE exit. The case in which the flow remains subsonic will be considered first. The Mach number at the inlet to the RWE is assumed to be the undisturbed local Mach number behind the shock. Using the definition of stagnation pressure, P_{0i} and solving for the flow Mach number at the RWE inlet, M_i yields Equation 6.

$$M_i = \left[\left(\frac{2}{\gamma - 1}\right)\left[\left(\frac{P_{0i}}{p_i}\right)^{\frac{\gamma - 1}{\gamma}} - 1\right]\right]^{0.5} \tag{6}$$

If p_i and P_{0i} are the static and stagnation pressures, respectively, at the RWE inlet, then the Mach number at the inlet, M_i, can be calculated using Equation 6.

For isentropic flow, the stagnation pressure remains constant through the converging nozzle. Therefore, the stagnation pressure at the RWE exit, P_{0e}, equals P_{0i} (the known inlet stagnation pressure). For the rarefaction wave to be eliminated, the static pressure at the RWE exit must match the atmospheric pressure, p_∞, which is also known. Therefore, Equation 6 can be modified and used to determine the Mach number at the RWE exit, M_e as a function of P_{0i} and p_∞ as shown in Equation 7.

$$M_e = \left[\left(\frac{2}{\gamma - 1}\right)\left[\left(\frac{P_{0i}}{p_\infty}\right)^{\frac{\gamma - 1}{\gamma}} - 1\right]\right]^{0.5} \tag{7}$$

The relationship between Mach number and area ratio for an isentropic flow is given by Equation 8.

$$\frac{A_e}{A_i} = \left(\frac{M_i}{M_e}\right) \left[\frac{2 + (\gamma - 1)M_e^2}{2 + (\gamma - 1)M_i^2}\right]^{\frac{\gamma+1}{2(\gamma-1)}} \tag{8}$$

In Equation 8, A_i is the known inlet area to the RWE. A_e is the RWE exit area that will produce a match between the static pressure of the exiting flow and the atmosphere.

For a decaying blast wave simulation, the static and stagnation pressures at the inlet of the RWE change with time. As long as these pressure histories are known, an RWE open area history can be found by repeated calculations using Equations 6, 7, and 8.

For the second case, in which the ratio p_∞/P_{0i} is less than 0.528282, the flow at the exit plane of the RWE is sonic, thus $M_e = 1.0$. Equation 6 is still used to calculate M_i, but since M_e is known Equation 7 is not used. In this case, Equation 8 simplifies to Equation 9.

$$\frac{A_e}{A_i} = M_i \left[\frac{\gamma + 1}{2 + (\gamma - 1)M_i^2}\right]^{\frac{\gamma+1}{2(\gamma-1)}} \tag{9}$$

For a decaying blast wave simulation with an initial p_∞/P_{0i} ratio less than 0.528282, Equation 9 is used from the start of the calculation until the value of this ratio reaches 0.528282. When this point is reached, Equation 8 is used for all subsequent area ratio calculations in the history.

In reality, the RWE on the LB/TS is not a simple converging nozzle. Rather, it is an array of flat plates that are rotated with time to change the available exit flow area of the expansion tunnel. The rotation of the plates is accomplished by driving a rack to which the plates are connected. The motion of the rack is induced by a hydraulic actuator, controlled by computer controlled servo. Based on the design of the RWE, a table of values relating the open area ratio, plate angle, rack position and servo voltage were derived. This table is incorporated into a computer program that generates a command signal for the RWE which is executed during a test.

To create an RWE motion command signal for a given test, it is necessary to have a prediction of the decaying blast history in the expansion section. These predictions are obtained through numerical simulation of flow in the facility with CFD codes. The code which was developed for generating LB/TS RWE functions is presently designed to read station history data generated by the BRL-Q1D code[6] or the SHARC code.[7] The code can be easily modified to accept input from other sources.

The LB/TS RWE function generation code is written in Fortran 77 and interactively prompts the user for pertinent information. A listing of this program is provided in Appendix E.

7. Passive RWE Setting

Sometimes, it is useful to obtain the initial RWE open area ratio required to eliminate the rarefaction resulting from a step shock of a given amplitude. This is often used for tests that may occur when the RWE area does not change with time. An interactive computer program was developed for this purpose. It employs the same theory as described above, but only applied to the incident shock. This program is listed in Appendix F.

Intentionally Left Blank

References

1. Glasstone, S. and P. Dolan - Editors. "The Effects of Nuclear Weapons." Department of Army Pamphlet No. 50-3, HQ, Department of Army. March 1977.

2. Schraml, S. "Performance Predictions for the Large Blast/Thermal Simulator Based on Experimental and Computational Results." U.S. Army Ballistic Research Laboratory Technical Report BRL-TR-3232. Aberdeen Proving Ground, MD. May 1991.

3. Smiley, R., J. Ruetenik and M. Tomayko. "Reflect-4 Code Computations of 40 kT Nuclear Blast Waves Reflected from the Ground." DNA-TR-81-203, Defense Nuclear Agency, Washington, DC. November 1992.

4. Zucrow, M. and J. Hoffman. "Gas Dynamics - Volume I." John Wiley & Sons, Inc. ISBN 0-471-98440-X. 1976.

5. Schraml S. and R. Pearson. "Small Scale Shock Tube Experiments Using a Computer Controlled Active Rarefaction Wave Eliminator." U.S. Army Ballistic Research Laboratory Technical Report BRL-TR-3149. Aberdeen Proving Ground, MD. September 1990.

6. Opalka, K. and A. Mark. "The BRL-Q1D Code: A Tool for the Numerical Simulation of Flows in Shock Tubes with Variable Cross-Sectional Areas." U.S. Army Ballistic Research Laboratory Technical Report BRL-TR-2763. Aberdeen Proving Ground, MD. October 1986.

7. Hikida, S., R. Bell, and C. Needham. "The SHARC Codes: Documentation and Sample Problems." S-Cubed Technical Report SSS-R-89-9878. September 1988.

Intentionally Left Blank

Appendix A: BLAST Program Listing and Usage

A sample input file for the BLAST program follows:

```
82.74    31.85    84.9439    288.71    100
```

in which the data describe, from left to right, the peak static overpressure at the observation point in kilopascals, the yield of the weapon in kilotons, the ambient pressure in kilopascals, the ambient temperature in Kelvins and the number of records desired in the resulting waveform. The resulting output is listed below:

```
PRESSURE HISTORY FOR A 12.0-PSI/0031-KT IDEAL BLAST WAVE WITH 193.23 METRE HOB
      AT AMBIENT CONDITIONS OF (P=)  84.94 KPA AND (T=) 288.71 KELVIN
RANGE FROM GROUND ZERO =  926.69 METRE ** SHOCK ARRIVAL TIME =  0.8022 SECONDS
==============================================================================
    SIDE-ON OVERPRESSURE HISTORY               DYNAMIC PRESSURE HISTORY
PEAK SIDE-ON OVERPRESSURE =  82.74 KPA    PEAK DYNAMIC PRESSURE =  25.33 KPA
    TIME        PRESSURE    IMPULSE           TIME        PRESSURE    IMPULSE
---------------------------------------  ---------------------------------------
```

0.00000E+00	0.10000E+01	0.00000E+00	0.00000E+00	0.10000E+01	0.00000E+00
0.81026E-02	0.97276E+00	0.66128E+00	0.13409E-01	0.93091E+00	0.32788E+00
0.16205E-01	0.94617E+00	0.13045E+01	0.26818E-01	0.86649E+00	0.63310E+00
0.24308E-01	0.92020E+00	0.19301E+01	0.40227E-01	0.80646E+00	0.91718E+00
0.32410E-01	0.89485E+00	0.25385E+01	0.53636E-01	0.75049E+00	0.11816E+01
0.40513E-01	0.87010E+00	0.31301E+01	0.67045E-01	0.69834E+00	0.14276E+01
0.48615E-01	0.84594E+00	0.37054E+01	0.80454E-01	0.64973E+00	0.16565E+01
0.56718E-01	0.82236E+00	0.42646E+01	0.93863E-01	0.60444E+00	0.18695E+01
0.64821E-01	0.79934E+00	0.48082E+01	0.10727E+00	0.56224E+00	0.20676E+01
0.72923E-01	0.77687E+00	0.53365E+01	0.12068E+00	0.52293E+00	0.22519E+01
0.81026E-01	0.75494E+00	0.58500E+01	0.13409E+00	0.48630E+00	0.24232E+01
0.89128E-01	0.73353E+00	0.63489E+01	0.14750E+00	0.45218E+00	0.25826E+01
0.97231E-01	0.71264E+00	0.68337E+01	0.16091E+00	0.42040E+00	0.27308E+01
0.10533E+00	0.69225E+00	0.73046E+01	0.17432E+00	0.39080E+00	0.28685E+01
0.11344E+00	0.67236E+00	0.77620E+01	0.18773E+00	0.36324E+00	0.29966E+01
0.12154E+00	0.65294E+00	0.82063E+01	0.20114E+00	0.33757E+00	0.31156E+01
0.12964E+00	0.63400E+00	0.86377E+01	0.21454E+00	0.31367E+00	0.32261E+01
0.13774E+00	0.61552E+00	0.90565E+01	0.22795E+00	0.29143E+00	0.33289E+01
0.14585E+00	0.59749E+00	0.94631E+01	0.24136E+00	0.27072E+00	0.34244E+01
0.15395E+00	0.57989E+00	0.98578E+01	0.25477E+00	0.25144E+00	0.35130E+01
0.16205E+00	0.56273E+00	0.10241E+02	0.26818E+00	0.23350E+00	0.35954E+01
0.17015E+00	0.54599E+00	0.10612E+02	0.28159E+00	0.21680E+00	0.36718E+01
0.17826E+00	0.52966E+00	0.10973E+02	0.29500E+00	0.20127E+00	0.37428E+01
0.18636E+00	0.51373E+00	0.11323E+02	0.30841E+00	0.18682E+00	0.38087E+01
0.19446E+00	0.49819E+00	0.11662E+02	0.32182E+00	0.17337E+00	0.38699E+01
0.20256E+00	0.48304E+00	0.11991E+02	0.33523E+00	0.16087E+00	0.39266E+01
0.21067E+00	0.46826E+00	0.12310E+02	0.34864E+00	0.14924E+00	0.39793E+01
0.21877E+00	0.45385E+00	0.12619E+02	0.36204E+00	0.13842E+00	0.40281E+01
0.22687E+00	0.43980E+00	0.12918E+02	0.37545E+00	0.12836E+00	0.40735E+01
0.23497E+00	0.42610E+00	0.13209E+02	0.38886E+00	0.11901E+00	0.41155E+01
0.24308E+00	0.41274E+00	0.13490E+02	0.40227E+00	0.11032E+00	0.41544E+01
0.25118E+00	0.39972E+00	0.13762E+02	0.41568E+00	0.10224E+00	0.41905E+01
0.25928E+00	0.38702E+00	0.14026E+02	0.42909E+00	0.94735E-01	0.42239E+01
0.26738E+00	0.37464E+00	0.14281E+02	0.44250E+00	0.87760E-01	0.42549E+01
0.27549E+00	0.36258E+00	0.14528E+02	0.45591E+00	0.81279E-01	0.42836E+01

0.28359E+00	0.35082E+00	0.14767E+02	0.46932E+00	0.75259E-01	0.43102E+01
0.29169E+00	0.33936E+00	0.14999E+02	0.48273E+00	0.69668E-01	0.43348E+01
0.29980E+00	0.32819E+00	0.15223E+02	0.49614E+00	0.64477E-01	0.43576E+01
0.30790E+00	0.31731E+00	0.15439E+02	0.50954E+00	0.59656E-01	0.43787E+01
0.31600E+00	0.30670E+00	0.15648E+02	0.52295E+00	0.55181E-01	0.43982E+01
0.32410E+00	0.29637E+00	0.15850E+02	0.53636E+00	0.51028E-01	0.44162E+01
0.33221E+00	0.28630E+00	0.16046E+02	0.54977E+00	0.47173E-01	0.44329E+01
0.34031E+00	0.27650E+00	0.16234E+02	0.56318E+00	0.43597E-01	0.44483E+01
0.34841E+00	0.26694E+00	0.16416E+02	0.57659E+00	0.40280E-01	0.44626E+01
0.35651E+00	0.25764E+00	0.16592E+02	0.59000E+00	0.37203E-01	0.44757E+01
0.36462E+00	0.24857E+00	0.16762E+02	0.60341E+00	0.34350E-01	0.44879E+01
0.37272E+00	0.23975E+00	0.16926E+02	0.61682E+00	0.31704E-01	0.44991E+01
0.38082E+00	0.23115E+00	0.17083E+02	0.63023E+00	0.29252E-01	0.45094E+01
0.38892E+00	0.22278E+00	0.17236E+02	0.64363E+00	0.26980E-01	0.45190E+01
0.39703E+00	0.21463E+00	0.17382E+02	0.65704E+00	0.24875E-01	0.45278E+01
0.40513E+00	0.20670E+00	0.17523E+02	0.67045E+00	0.22924E-01	0.45359E+01
0.41323E+00	0.19897E+00	0.17659E+02	0.68386E+00	0.21118E-01	0.45434E+01
0.42133E+00	0.19146E+00	0.17790E+02	0.69727E+00	0.19446E-01	0.45503E+01
0.42944E+00	0.18414E+00	0.17916E+02	0.71068E+00	0.17898E-01	0.45566E+01
0.43754E+00	0.17702E+00	0.18037E+02	0.72409E+00	0.16465E-01	0.45624E+01
0.44564E+00	0.17009E+00	0.18154E+02	0.73750E+00	0.15140E-01	0.45678E+01
0.45374E+00	0.16334E+00	0.18265E+02	0.75091E+00	0.13914E-01	0.45727E+01
0.46185E+00	0.15678E+00	0.18373E+02	0.76432E+00	0.12781E-01	0.45773E+01
0.46995E+00	0.15040E+00	0.18476E+02	0.77773E+00	0.11733E-01	0.45814E+01
0.47805E+00	0.14419E+00	0.18574E+02	0.79113E+00	0.10765E-01	0.45853E+01
0.48615E+00	0.13815E+00	0.18669E+02	0.80454E+00	0.98699E-02	0.45888E+01
0.49426E+00	0.13228E+00	0.18760E+02	0.81795E+00	0.90437E-02	0.45920E+01
0.50236E+00	0.12657E+00	0.18846E+02	0.83136E+00	0.82810E-02	0.45949E+01
0.51046E+00	0.12102E+00	0.18929E+02	0.84477E+00	0.75770E-02	0.45976E+01
0.51856E+00	0.11562E+00	0.19009E+02	0.85818E+00	0.69275E-02	0.46001E+01
0.52667E+00	0.11037E+00	0.19085E+02	0.87159E+00	0.63285E-02	0.46023E+01
0.53477E+00	0.10527E+00	0.19157E+02	0.88500E+00	0.57763E-02	0.46044E+01
0.54287E+00	0.10031E+00	0.19226E+02	0.89841E+00	0.52675E-02	0.46063E+01
0.55097E+00	0.95495E-01	0.19291E+02	0.91182E+00	0.47988E-02	0.46080E+01
0.55908E+00	0.90814E-01	0.19354E+02	0.92523E+00	0.43672E-02	0.46095E+01
0.56718E+00	0.86267E-01	0.19413E+02	0.93863E+00	0.39700E-02	0.46109E+01
0.57528E+00	0.81851E-01	0.19470E+02	0.95204E+00	0.36047E-02	0.46122E+01
0.58338E+00	0.77561E-01	0.19523E+02	0.96545E+00	0.32688E-02	0.46134E+01
0.59149E+00	0.73395E-01	0.19574E+02	0.97886E+00	0.29602E-02	0.46144E+01
0.59959E+00	0.69351E-01	0.19621E+02	0.99227E+00	0.26767E-02	0.46154E+01
0.60769E+00	0.65424E-01	0.19667E+02	0.10057E+01	0.24165E-02	0.46163E+01
0.61580E+00	0.61613E-01	0.19709E+02	0.10191E+01	0.21778E-02	0.46171E+01
0.62390E+00	0.57913E-01	0.19749E+02	0.10325E+01	0.19589E-02	0.46178E+01
0.63200E+00	0.54324E-01	0.19787E+02	0.10459E+01	0.17585E-02	0.46184E+01
0.64010E+00	0.50841E-01	0.19822E+02	0.10593E+01	0.15749E-02	0.46189E+01
0.64821E+00	0.47463E-01	0.19855E+02	0.10727E+01	0.14070E-02	0.46195E+01
0.65631E+00	0.44187E-01	0.19886E+02	0.10861E+01	0.12535E-02	0.46199E+01
0.66441E+00	0.41009E-01	0.19914E+02	0.10995E+01	0.11133E-02	0.46203E+01
0.67251E+00	0.37929E-01	0.19941E+02	0.11130E+01	0.98539E-03	0.46207E+01
0.68062E+00	0.34943E-01	0.19965E+02	0.11264E+01	0.86875E-03	0.46210E+01
0.68872E+00	0.32049E-01	0.19988E+02	0.11398E+01	0.76251E-03	0.46213E+01
0.69682E+00	0.29244E-01	0.20008E+02	0.11532E+01	0.66585E-03	0.46215E+01
0.70492E+00	0.26527E-01	0.20027E+02	0.11666E+01	0.57800E-03	0.46217E+01
0.71303E+00	0.23896E-01	0.20044E+02	0.11800E+01	0.49826E-03	0.46219E+01

0.72113E+00	0.21347E-01	0.20059E+02	0.11934E+01	0.42597E-03	0.46221E+01
0.72923E+00	0.18880E-01	0.20073E+02	0.12068E+01	0.36052E-03	0.46222E+01
0.73733E+00	0.16492E-01	0.20084E+02	0.12202E+01	0.30137E-03	0.46223E+01
0.74544E+00	0.14180E-01	0.20095E+02	0.12336E+01	0.24798E-03	0.46224E+01
0.75354E+00	0.11944E-01	0.20103E+02	0.12470E+01	0.19989E-03	0.46225E+01
0.76164E+00	0.97812E-02	0.20111E+02	0.12605E+01	0.15665E-03	0.46225E+01
0.76974E+00	0.76895E-02	0.20117E+02	0.12739E+01	0.11785E-03	0.46226E+01
0.77785E+00	0.56672E-02	0.20121E+02	0.12873E+01	0.83119E-04	0.46226E+01
0.78595E+00	0.37128E-02	0.20124E+02	0.13007E+01	0.52111E-04	0.46226E+01
0.79405E+00	0.18242E-02	0.20126E+02	0.13141E+01	0.24503E-04	0.46226E+01
0.80215E+00	0.00000E+00	0.20127E+02	0.13275E+01	0.00000E+00	0.46226E+01

```
      PROGRAM BLAST
C
C  **   **   **   **   **   **   **   **   **   **   **   **   **   **
C
C**  THIS PROGRAM CALCULATES IDEAL PRESSURE HISTORIES USING THE
C    MODIFIED FRIEDLANDER EQUATION AND BLAST-WAVE PARAMETERS
C    PREDICTED BY THE REFLECT-4 CODE AND SCALED TO THE GIVEN
C    WEAPON YIELD BY SACHS' SCALING.
C    DATA FOR STATIC AND DYNAMIC PRESSURES AS WELL AS IMPULSES
C    ARE TABULATED VERSUS TIME.
C
C**  INPUT PARAMETER LIST
C      PSO   - SHOCK OVERPRESSURE, KILOPASCAL
C      YIELD - WEAPON YIELD, KILOTON
C      PAMB  - AMBIENT PRESSURE, KILOPASCAL
C      TAMB  - AMBIENT, ATMOSPHERIC TEMPERATURE, KELVIN
C      IMX   - NUMBER OF DATA LINES ON OUTPUT (MAX = 301)
C
C**  THIS PROGRAM CALLS SUBPROGRAMS DVDINT AND BASER4;
C
C**  THE FOLLOWING CONNOTATION IS USED IN THIS PROGRAM
C    G1 = GROUND RANGE,               G5 = POSITIVE-PHASE DURATION
C    G2 = SHOCK ARRIVAL TIME,         G6 = PEAK DYNAMIC PRESSURE
C    G3 = PEAK SIDE-ON OVERPRESSURE,  G7 = DYN-PRESSURE DECAY CONSTANT
C    G4 = STAT-PRESS DECAY CONSTANT,  G8 = DYNAMIC-PHASE DURATION
C
C  **   **   **   **   **   **   **   **   **   **   **   **   **   **
C
      DIMENSION G1(60),G2(60),G3(60),G4(60),G5(60),G6(60),G7(60),G8(60)
      COMMON/IDEAL/  PP(301),QI(301),QQ(301),SI(301),TT(301),TQ(301)
      COMMON/SANCA/ HI(60,8)
      EQUIVALENCE (G1(1),HI(1,1)),(G2(1),HI(1,2)),(G3(1),HI(1,3))
     +           ,(G4(1),HI(1,4)),(G5(1),HI(1,5)),(G6(1),HI(1,6))
     +           ,(G7(1),HI(1,7)),(G8(1),HI(1,8))
      PARAMETER (WT=40.,BH=208.48)
C
C
C**  (1) READ INPUT AND SELECT BLAST-WAVE PARAMETERS
C
   10 READ(*,*,END=99) PSO,YIELD,PAMB,TAMB,IMX
      S1 = PAMB/101.325
      S2 = (YIELD/WT/S1)**0.333333
      S3 = S2*SQRT(288.15/TAMB)
C
      PMX = PSO/S1
      CALL DVDINT (PMX,RNG,G3,G1,60,4,IER)
      RNG = RNG*S2
      CALL DVDINT (PMX,TAS,G3,G5,60,4,IER)
```

```
      TAS = TAS*S3
      CALL DVDINT (PMX,CI ,G3,G4,60,4,IER)
      CALL DVDINT (PMX,PPD,G3,G5,60,4,IER)
      PPD = PPD*S3
      CALL DVDINT (PMX,QMX,G3,G6,60,4,IER)
      QMX = QMX*S1
      CALL DVDINT (PMX,CIQ,G3,G7,60,4,IER)
      CALL DVDINT (PMX,QPD,G3,G8,60,4,IER)
      QPD = QPD*S3
C
C
C**  (2)  DEFINE PRESSURE AND IMPULSE VS TIME
C
      TT(1) = 0.
      TQ(1) = 0
      PP(1) = 1.
      QQ(1) = 1.
      SI(1) = 0.
      QI(1) = 0.
      DO 20 N=2,IMX-1,1
         TAU = REAL(N-1)/(IMX-1)
         TT(N) = PPD*TAU
         TQ(N) = QPD*TAU
         PP(N) = (1.-TAU)*EXP(-CI*TAU)
         QQ(N) = (1.-TAU)*EXP(-CIQ*TAU)
         SI(N) = SI(N-1) + 0.5*PSO*(PP(N-1)+PP(N))*PPD/(IMX-1)
         QI(N) = QI(N-1) + 0.5*QMX*(QQ(N-1)+QQ(N))*QPD/(IMX-1)
   20 CONTINUE
      TT(IMX) = PPD
      TQ(IMX) = QPD
      PP(IMX) = 0.
      QQ(IMX) = 0.
      SI(IMX) = SI(IMX-1) + 0.5*PSO*PP(IMX-1)*PPD/(IMX-1)
      QI(IMX) = QI(IMX-1) + 0.5*QMX*QQ(IMX-1)*QPD/(IMX-1)
C
C
C**  (3)  TABULATE DATA
C
      IWY = YIELD
      PSI = PSO/6.8947572
      HOB = BH*(YIELD/WT)**0.333333
      LP = 1
      L1 = 1
      L2 = 49
      IF (IMX.LT.49)  L2 = IMX
      WRITE(*,31) PSI,IWY,HOB,PAMB,TAMB,RNG,TAS
      WRITE(*,32) PSO,QMX
   30 WRITE(*,33) (TT(I),PP(I),SI(I),TQ(I),QQ(I),QI(I),I=L1,L2)
```

```
      IF (L2.GE.IMX)  GOTO 40
C
      LP = LP+1
      L1 = L2+1
      L2 = MIN0(IMX,L2+52)
      WRITE(*,34) LP
      WRITE(*,32) PSO,QMX
      GOTO 30
C
   40 CONTINUE
      GOTO 10
C
   99 STOP
C
   31 FORMAT(1H1/T5,'PRESSURE HISTORY FOR A',F5.1,'-PSI/',I4.4,
     +  '-KT IDEAL BLAST WAVE WITH ',F6.2,' METRE HOB'
     +  /T11,'AT AMBIENT CONDITIONS OF (P=) ',F6.2,' KPA AND (T=) ',
     +  F6.2,' KELVIN' /T5,'RANGE FROM GROUND ZERO =',F8.2,
     +  ' METRE ** SHOCK ARRIVAL TIME =',F8.4,' SECONDS' /T5,78('='))
C
   32 FORMAT (1H0,T9,'SIDE-ON OVERPRESSURE HISTORY',
     + T52,'DYNAMIC PRESSURE HISTORY'
     + /T4,'PEAK SIDE-ON OVERPRESSURE = ',F6.2,' KPA ',
     + T47,'PEAK DYNAMIC PRESSURE = ',F6.2,' KPA'
     + /T9,'TIME',T20,'PRESSURE   IMPULSE',T50,'TIME',
     + T61,'PRESSURE   IMPULSE' /T4,39('-'),T45,39('-'))
C
   33 FORMAT (T5,E11.5,2X,E11.5,2X,E11.5,4X,E11.5,2X,E11.5,2X,E11.5)
C
   34 FORMAT (1H1,T77,'PAGE',I2)
      END
```

```
      SUBROUTINE DVDINT (XI,YI,XD,DD,KT,N,IER)
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
C**  THIS SUBROUTINE DOES DIVIDED-DIFFERENCE INTERPOLATION. BE SURE
C    THAT THE ARGUMENTS HAVE BEEN TABULATED IN DESCENDING ORDER.
C    THE SUBROUTINE IS CALLED FROM SUBPROGRAM IMPULS.
C
C**  I/O PARAMETER LIST:
C (I)  XI  = ARGUMENT FOR WHICH FUNCTIONAL VALUE IS DESIRED.
C (O)  YI  = NAME OF THE RESULT.
C (I)  XD  = (1D) ARRAY OF X-VALUES.
C (I)  DD  = (1D) ARRAY OF FUNCTION VALUES.
C (I)  KT  = NUMBER OF VALUES IN XD AND DD ARRAYS.
C (I)  N   = NUMBER OF POINTS TO USE IN EACH INTERPOLATION.
C (O)  IER = ERROR FLAG SET = 111 WHEN ARGUMENT NOT IN TABLE.
C              AT LOWER END:  IF (XI.LT.XD(1))    YI = DD(1)
C              AT UPPER END:  IF (XI.GT.XD(KT))   YI = DD(KT)
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
      DIMENSION XD(KT), DD(KT), U1(16)
C
C
C**  1. INITIALIZE:
C    --------------
C
      IER = 0
      L1 = (N-1)/2
      L2 = N/2
      L3 = KT-L2+1
      L4 = L1+2
      L5 = KT-N
C
C
C**  2. FIND ENTRY INTO TABLE
C    ------------------------
C
      IF ((XI-2*XD(1)+XD(2)).GE.0..OR.(XI-2*XD(KT)+XD(KT-1)).LT.0.)
     + GOTO 100
C
   40 L5 = L5/2
      JI = L4+L5
      IF (XD(JI).GT.XI) L4 = JI
      IF (L5.GT.1)  GOTO 40
C
   50 IF (XI.GT.XD(L4).OR.L4.EQ.L3)  GOTO 60
C
```

```
      L4 = L4+1
      GOTO 50
C
C
C**  3. EXECUTE INTERPOLATION
C    ------------------------
C
   60 L4 = L4-1
      L5 = L4-L1
      DO 70 I=1,N
         U1(I) = DD(L5)
         L5 = L5+1
   70    CONTINUE
      L = (N+1)/2
      YI = U1(L)
      JI = L4
      K = L4+1
      JU = 1
      L2 = N-1
      U2 = 1.0
      DO 90 J=1,L2
         L5 = L4-L1
         L3 = N-J
         DO 80 I=1,L3
            M = L5+J
            U1(I) = (U1(I+1)-U1(I))/(XD(M)-XD(L5))
            L5 = L5+1
   80       CONTINUE
         IF (JU.EQ.1)  THEN
            U2 = U2*(XI-XD(JI))
            JU = 2
            JI = JI-1
         ELSE
            U2 = U2*(XI-XD(K))
            JU = 1
            K = K+1
            L = L-1
         ENDIF
         YI = YI+U2*U1(L)
   90    CONTINUE
      RETURN
C
C
C**  4. ERROR EXIT
C    -------------
C
  100 IER = 111
      IF (XI.LT.XD(1))  YI = DD(1)
```

```
      IF (XI.GT.XD(KT)) YI = DD(KT)
      PRINT 110, XI,XD(1),XD(KT)
      RETURN
C
  110 FORMAT (1H ,T7,'ARG. NOT IN TABLE  XI= ',E12.6,'  XD(1)= ',E12.6,
     + '  XD(KT)= ',E12.6,2X,6HDVDINT)
C
      END
```

```
      BLOCK DATA BASER4
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
C**  THIS SUBPROGRAM SPECIFIES THE REFERENCE DATA BASE FOR A 40 KT
C    NUCLEAR BLAST WAVE, BASED ON REFLECT-4 CODE COMPUTATIONS.
C
C    =================================================================
C**                      ARRAY DIMENSIONS: HI(60,8)
C    =================================================================
C
C**  ORGANIZATION OF DATA ARRAY:
C    COL (1) = GROUND RANGE IN METRE (GR)
C    COL (2) = TIME OF SHOCK ARRIVAL IN SECONDS (TA)
C    COL (3) = STATIC SHOCK OVERPRESSURE IN KILOPASCALS (OP)
C    COL (4) = STATIC-OVERPRESSURE DECAY CONSTANT (CIS)
C    COL (5) = POSITIVE-PHASE DURATION IN SECONDS (TP)
C    COL (6) = MAXIMUM DYNAMIC PRESSURE IN KILOPASCALS (DP)
C    COL (7) = DYNAMIC-PRESSURE DECAY CONSTANT (CIQ)
C    COL (8) = DYNAMIC-PRESSURE PHASE DURATION IN SECONDS (TQ)
C
C**  NOTICE:
C    THE IMPULSE MAY BE OBTAINED BY INTEGRATING THE MODIFIED
C    FRIEDLANDER EQUATION,  P(T) = PMAX*(1-T/PPD)*EXP(-CI*T/PPD)
C    WITH THE HELP OF A LIBRARY SUBROUTINE, OR
C    THE INTEGRATED EQUATION,  I = PMAX*PPD*(EXP(-CI)+CI-1)/CI**2
C    MAY BE USED.
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
      COMMON/SANCA/ HI(60,8)
C
      DATA(HI(01,J),J=1,8)/354.5, .2829, 690.21, 2.7355, 0.302, 1177.1,
     +   77.279, 4.455/
      DATA(HI(02,J),J=1,8)/365.1, .2949, 655.16, 2.7213, 0.311, 1010.6,
     +   69.516, 4.468/
      DATA(HI(03,J),J=1,8)/377.3, .3092, 620.34, 2.7502, 0.322, 840.88,
     +   61.402, 4.482/
      DATA(HI(04,J),J=1,8)/390.2, .3246, 585.82, 2.7786, 0.334, 692.83,
     +   54.640, 4.498/
      DATA(HI(05,J),J=1,8)/402.6, .3397, 550.91, 2.7309, 0.344, 614.83,
     +   52.992, 4.518/
      DATA(HI(06,J),J=1,8)/415.5, .3559, 517.24, 2.6845, 0.355, 555.50,
     +   51.935, 4.550/
      DATA(HI(07,J),J=1,8)/431.6, .3766, 482.97, 2.6743, 0.369, 496.75,
     +   52.243, 4.647/
      DATA(HI(08,J),J=1,8)/449.6, .4004, 447.95, 2.6683, 0.386, 439.09,
     +   52.900, 4.713/
```

```
 DATA(HI(09,J),J=1,8)/468.3, .4260, 413.42, 2.6253, 0.403, 384.64,
+   52.368, 4.733/
 DATA(HI(10,J),J=1,8)/488.0, .4537, 379.87, 2.5981, 0.422, 332.69,
+   50.772, 4.745/
 DATA(HI(11,J),J=1,8)/511.8, .4884, 345.46, 2.5545, 0.443, 283.18,
+   49.122, 4.765/
 DATA(HI(12,J),J=1,8)/538.1, .5283, 311.06, 2.4672, 0.465, 237.32,
+   47.365, 4.824/
 DATA(HI(13,J),J=1,8)/553.4, .5521, 293.50, 2.4221, 0.478, 214.99,
+   46.707, 4.893/
 DATA(HI(14,J),J=1,8)/570.8, .5798, 275.08, 2.3628, 0.492, 192.38,
+   45.284, 4.908/
 DATA(HI(15,J),J=1,8)/583.8, .6010, 262.14, 2.3268, 0.504, 177.04,
+   44.119, 4.911/
 DATA(HI(16,J),J=1,8)/599.0, .6260, 248.04, 2.2760, 0.517, 160.83,
+   42.725, 4.910/
 DATA(HI(17,J),J=1,8)/614.6, .6524, 234.49, 2.2274, 0.531, 145.81,
+   41.329, 4.909/
 DATA(HI(18,J),J=1,8)/631.9, .6822, 220.59, 2.1616, 0.545, 131.12,
+   39.927, 4.913/
 DATA(HI(19,J),J=1,8)/650.9, .7155, 206.96, 2.0956, 0.560, 117.14,
+   38.719, 4.947/
 DATA(HI(20,J),J=1,8)/662.2, .7357, 199.49, 2.0651, 0.570, 109.73,
+   37.942, 4.947/
 DATA(HI(21,J),J=1,8)/672.1, .7534, 193.37, 2.0510, 0.580, 103.80,
+   37.117, 4.928/
 DATA(HI(22,J),J=1,8)/683.6, .7744, 186.61, 2.0314, 0.591, 97.395,
+   36.228, 4.904/
 DATA(HI(23,J),J=1,8)/697.4, .7998, 179.04, 2.0087, 0.604, 90.425,
+   35.156, 4.874/
 DATA(HI(24,J),J=1,8)/710.4, .8239, 172.40, 1.9811, 0.615, 84.475,
+   17.151, 2.501/
 DATA(HI(25,J),J=1,8)/723.9, .8494, 165.87, 1.9443, 0.625, 78.779,
+   13.492, 2.044/
 DATA(HI(26,J),J=1,8)/739.5, .8793, 158.79, 1.9132, 0.638, 72.788,
+   12.173, 1.904/
 DATA(HI(27,J),J=1,8)/757.4, .9138, 151.30, 1.8806, 0.653, 66.652,
+   11.112, 1.800/
 DATA(HI(28,J),J=1,8)/773.5, .9453, 145.03, 1.8755, 0.670, 61.701,
+   10.401, 1.736/
 DATA(HI(29,J),J=1,8)/793.9, .9858, 137.68, 1.8733, 0.692, 56.082,
+    9.001, 1.571/
 DATA(HI(30,J),J=1,8)/815.1, 1.028, 130.66, 1.8730, 0.715, 50.925,
+   7.9576, 1.454/
 DATA(HI(31,J),J=1,8)/835.2, 1.069, 124.55, 1.8822, 0.738, 46.615,
+   7.3794, 1.400/
 DATA(HI(32,J),J=1,8)/860.6, 1.122, 117.47, 1.8623, 0.761, 41.824,
+   6.9488, 1.374/
```

```
      DATA(HI(33,J),J=1,8)/891.7, 1.187, 109.70, 1.7941, 0.780, 36.818,
     +  6.5063, 1.350/
      DATA(HI(34,J),J=1,8)/918.1, 1.243, 103.77, 1.7681, 0.801, 33.184,
     +  6.2561, 1.347/
      DATA(HI(35,J),J=1,8)/955.3, 1.324, 96.271, 1.7064, 0.824, 28.834,
     +  5.9932, 1.353/
      DATA(HI(36,J),J=1,8)/992.2, 1.405, 89.728, 1.6390, 0.843, 25.249,
     +  5.6584, 1.341/
      DATA(HI(37,J),J=1,8)/1039., 1.509, 82.482, 1.5788, 0.870, 21.532,
     +  5.3987, 1.348/
      DATA(HI(38,J),J=1,8)/1089., 1.624, 75.684, 1.5071, 0.895, 18.285,
     +  5.1998, 1.365/
      DATA(HI(39,J),J=1,8)/1149., 1.762, 68.844, 1.4393, 0.924, 15.265,
     +  4.9788, 1.382/
      DATA(HI(40,J),J=1,8)/1217., 1.922, 62.232, 1.3685, 0.955, 12.583,
     +  4.7662, 1.398/
      DATA(HI(41,J),J=1,8)/1260., 2.025, 58.599, 1.3245, 0.973, 11.211,
     +  4.5368, 1.388/
      DATA(HI(42,J),J=1,8)/1306., 2.138, 55.034, 1.2847, 0.993,  9.935,
     +  4.4825, 1.413/
      DATA(HI(43,J),J=1,8)/1357., 2.261, 51.580, 1.2440, 1.013,  8.763,
     +  4.3289, 1.416/
      DATA(HI(44,J),J=1,8)/1411., 2.395, 48.243, 1.2025, 1.034,  7.701,
     +  4.1743, 1.423/
      DATA(HI(45,J),J=1,8)/1475., 2.554, 44.809, 1.1604, 1.058,  6.674,
     +  4.0918, 1.447/
      DATA(HI(46,J),J=1,8)/1548., 2.737, 41.362, 1.1147, 1.083,  5.716,
     +  3.9617, 1.461/
      DATA(HI(47,J),J=1,8)/1633., 2.953, 37.887, 1.0695, 1.112,  4.819,
     +  3.7906, 1.469/
      DATA(HI(48,J),J=1,8)/1732., 3.209, 34.419, 1.0204, 1.143,  3.992,
     +  3.6360, 1.484/
      DATA(HI(49,J),J=1,8)/1849., 3.514, 30.992, .97131, 1.178,  3.254,
     +  3.4870, 1.505/
      DATA(HI(50,J),J=1,8)/1990., 3.883, 27.614, .91900, 1.216,  2.599,
     +  3.3651, 1.527/
      DATA(HI(51,J),J=1,8)/2166., 4.353, 24.194, .86311, 1.261,  1.999,
     +  3.1368, 1.542/
      DATA(HI(52,J),J=1,8)/2400., 4.983, 20.684, .80528, 1.315,  1.469,
     +  2.9593, 1.574/
      DATA(HI(53,J),J=1,8)/2704., 5.812, 17.292, .73804, 1.378,  1.034,
     +  2.8287, 1.611/
      DATA(HI(54,J),J=1,8)/3153., 7.054, 13.790, .66684, 1.461,  0.662,
     +  2.6993, 1.676/
      DATA(HI(55,J),J=1,8)/3823., 8.933, 10.349, .55874, 1.571,  0.372,
     +  2.2476, 1.726/
      DATA(HI(56,J),J=1,8)/5068., 12.467, 6.902, .43014, 1.753,  0.165,
     +  1.9634, 1.905/
```

```
      DATA(HI(57,J),J=1,8)/5924., 14.922, 5.523, .36010, 1.874,  0.110,
     +   1.9454, 1.960/
      DATA(HI(58,J),J=1,8)/7254., 18.756, 4.137, .27151, 2.072,  0.062,
     +   1.6035, 2.113/
      DATA(HI(59,J),J=1,8)/8250., 21.639, 3.447, .22038, 2.240,  0.041,
     +   1.0797, 2.305/
      DATA(HI(60,J),J=1,8)/9110., 24.138, 2.999, .18924, 2.407,  0.034,
     +   1.3186, 2.443/
C
      END
```

Appendix B: YIELD Program Listing and Usage

A sample input for the YIELD program follows:

```
1      84.944      288.71      82.738      4.62      4.62
```

in which the data describe, from left to right, a case number, the ambient pressure in kilopascals, the ambient temperature in Kelvins, the peak static overpressure at the observation point in kilopascals, the static overpressure impulse at the observation point in kilopascal-seconds and the dynamic pressure impulse at the observation point in kilopascal-seconds. The resulting output is listed below:

```
      SIDE-ON  SIDE-ON  SIDE-ON  DYNAMIC  DYNAMIC
       OVERP.  IMPULSE    YIELD  IMPULSE    YIELD
        (kPa)  (kPa-s)     (kT)  (kPa-s)     (kT)

 1      82.74     4.62     0.39     4.62    31.85
```

The case described here illustrates the fact that the static and dynamic pressure impulse provided in the input are not required to correspond to the same weapon. Any two independent impulse values may be used, and the program will produce a weapon yield for each.

The YIELD program also employs the DVDINT and BASER4 routines that were listed in Appendix A.

```
C  ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** *****
C
      PROGRAM MAIN
C
C  ***** ***** ***** ***** ***** ***** ***** ***** ***** ***** *****
C
      COMMON/RSLT/ PPF,WYQ,WYS,PPD,PSO,QIM,QPD,QPF,QS,QSF,SIM,TAR
      COMMON/AREV/ NMAX,MMAX,JMAX,JM1,DT,TAU,SMU,GAMA,GAMI,CN
     *                ,PRAT,TRAT,PAMB,TAMB,TICK,NXK
      LOGICAL ERROR
      REAL YIELD(20,5)
      INTEGER I,J,K
C
      OPEN(1,FILE='yield.in',STATUS='OLD')
      OPEN(2,FILE='yield.out',STATUS='NEW')
C
      WRITE (2,200)
  200 FORMAT (8X,'SIDE-ON',3X,'SIDE-ON',3X,'SIDE-ON',
     +        3X,'DYNAMIC',3X,'DYNAMIC'/
     +        9X,'OVERP.',3X,'IMPULSE',5X,'YIELD',
     +        3X,'IMPULSE',5X,'YIELD'/
     +       10X,'(kPa)',3X,'(kPa-s)',6X,'(kT)',
     +        3X,'(kPa-s)',6X,'(kT)'/)
C
      I = 0
    2 READ (1,*,END=5) ITEST,PAMB,TAMB,PSO,SIM,QIM
      I = I + 1
      CALL IMPULS(1,ERROR)
      WRITE (2,250) ITEST,PSO,SIM,WYS,QIM,WYQ
  250 FORMAT (1X,I4,5F10.2)
      GOTO 2
C
    5 CONTINUE
      CLOSE(1)
      CLOSE(2)
      STOP
      END
```

```
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
      SUBROUTINE IMPULS (I,ERR)
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
C**  THIS SUBROUTINE FINDS THE TIME OF SHOCK ARRIVAL AT XSTA(I),TAR,
C    THE PEAK SHOCK OVERPRESSURE, PSO, AND THE POSITIVE-PHASE DURATION,
C    PPD.  IT DETERMINES THE STATIC AND DYNAMIC IMPULSES, SIM AND QIM,
C    USING SIMPSON'S RULE OF INTEGRATION AND CALCULATES THE EQUIVALENT
C    NUCLEAR WEAPON YIELD FOR BOTH IMPULSES, WYS AND WYQ.
C
C**  I/O PARAMETER LIST:
C      I    - INDEX OF OUTPUT STATION
C      ERR - ERROR FLAG FOR MMAX < 33
C
C**  THIS SUBROUTINE CALLS SUBPROGRAM DVDINT, AND
C      IS CALLED FROM THE MAIN PROGRAM.
C
C**  THE FOLLOWING CONNOTATION IS USED IN THIS SUBPROGRAM
C    G1 = GROUND RANGE,                G2 = SHOCK-ARRIVAL TIME,
C    G3 = STATIC OVERPRESSURE,         G4 = STATIC-OVERPRESSURE IMPULSE
C    G5 = POSITIVE-PHASE DURATION,     G6 = PEAK DYNAMIC PRESSURE
C    G7 = DYNAMIC-PRESSURE IMPULSE,    G8 = DYNAMIC-PHASE DURATION
C
C  **    **    **    **    **    **    **    **    **    **    **    **    **    **
C
      LOGICAL ERR
      DIMENSION G1(60),G2(60),G3(60),G4(60),G5(60),G6(60),G7(60),G8(60)
      COMMON/SANCA/ HI(60,8)
      COMMON/RSLT/ PPF,WYQ,WYS,PPD,PSO,QIM,QPD,QPF,QS,QSF,SIM,TAR
      EQUIVALENCE (G1(1),HI(1,1)),(G2(1),HI(1,2)),(G3(1),HI(1,3))
     +           ,(G4(1),HI(1,4)),(G5(1),HI(1,5)),(G6(1),HI(1,6))
     +           ,(G7(1),HI(1,7)),(G8(1),HI(1,8))
      COMMON/AREV/ NMAX,MMAX,JMAX,JM1,DT,TAU,SMU,GAMA,GAMI,CN
     +            ,PRAT,TRAT,PAMB,TAMB,TICK,NXK
      COMMON/LENG/ XDPH,DTHR,XDR4,XSTA(6),DDR4,DVNL,RWEL,REFL,DREF
     +            ,AREF,VOLD,PI4TH
      COMMON/STATN/ POUT(6000,6),SIMP(6000),PDYN(6000,6),QIMP(6000)
      COMMON/STATO/ VOUT(6000,6),ROUT(6000,6),TAW(6000),ARJX(6000)
      PARAMETER (KT=40,DIFF=.1E-11)
      CONTINUE
C
C
C**  (1) INITIALIZE VARIABLES
C
      C2  = 0.
```

```
      PPD = 0.                                                          024180
      PPF = 0.                                                          024190
      QS  = 0.3                                                         024210
      QPD = 0.                                                          024220
      QPF = 0.                                                          024230
      QSF = 0.                                                          024240
      TAR = 0.                                                          024250
      WYQ = 0.                                                          024260
      WYS = -1.00                                                       024270
C                                                                       024340
C**  COMPUTE WEAPON YIELD                                               025070
C                                                                       025080
      S1 = PAMB/101.325                                                 025090
      S2 = (1./S1)**0.333333 *SQRT(288.15/TAMB)                         025100
      S3 = S2*S1                                                        025110
      C2 = PSO/S1                                                       025120
C**      SIDE-ON OVERPRESSURE IMPULSE(S3):                              025130
      CALL DVDINT (C2,RTX,G3,G4,60,4,IER)                               025140
      IF (IER.LE.100)  WYS = KT*(SIM/RTX/S3)**3                         025150
      IF (IER.GT.100)  WYS = 0.0                                        025160
C**      DYNAMIC-PRESSURE IMPULSE(S3):
      CALL DVDINT (C2,RTX,G3,G7,60,4,IER)
      IF (IER.LE.100)  WYQ = KT*(QIM/RTX/S3)**3
      IF (IER.GT.100)  WYQ = 0
      RETURN                                                            025310
      END                                                               025320
```

Intentionally Left Blank

Appendix C: SHOCK Program Listing

```
      Program shock
c
c-----------------------------------------------------------------
c This program calculates air shock parameters using shock velocity,
c ambient temperature and ambient pressure.
c
c The program will alternatively calculate shock parameters using
c shock overpressure and ambient temperature and ambient pressure.
c
      Real M2F
      Print*,"Type in ambient temperature,T1, in deg.C."
      Read*,T1
      Print*,"Type in ambient pressure,P1, in kPa."
      Read*,P1
      Print*,"Is shock calculation to be based on velocity? If Yes,type
     + 1. If no,type 0"
      Read*,V
c
      If(V.EQ.1)Then
c
      Print*,"type in baseline length,D, in meters."
      Read*,D
      Print*,"type in elapsed time,TM, in seconds."
      Read*,TM
      W1=D/TM
      A1=331.6+.6069*T1
      W11=W1/A1
      P21=(7*W11**2-1)/6
c
      Else
c
      Print*,"Type in the shock overpressure in kPa."
      Read*,PS
      A1=331.6+.6069*T1
      P21=(PS+P1)/P1
      W11=((6*P21+1)/7)**0.5
      W1=W11*A1
c
      Endif
c
      PS =(P21-1)*P1
      U21 =(W11-1/W11)/1.2
      U2=U21*A1
      A21=((P21*(6+P21))/(1+6*P21))**.5
      A2=A21*A1
      M2F=U21/A21
      T21=A21**2
      G21=P21/T21
```

```
      Q21=.7*P21*(M2F)**2
      Q2=Q21*P1
cTotal head pressure ratio for M2F<1 is:
      P021A=P21*(1+.2*M2F**2)**3.5
      PstagA=(P021A-1)*P1
cTotal head pressure ratio for M2F>1 is:
      P021B=166.92*(M2F**2/(7*M2F**2-1))**2.5*P21
      PstagB=(P021B-1)*P1
      P12=1/P21
      P52=(8-P12)/(6*P12+1)
      T52=P52*(6+P52)/(6*P52+1)
      W21=W11*(2*P21+5)/(6*P21+1)
      DP5=(P52*P21-1)*P1
c
      Print*, "Baseline length,D, =",d,"meters"
      Print*, "shock travel time,tm, =",tm,"seconds"
      Print*, "ambient temperature,T1,=",t1, "deg.c"
      Print*,"ambient pressure,P1,=",p1,"kpa"
      print*,"ambient sound speed,A1,=",a1,"m/s"
      print*
      print*, "shock wave speed,W1,=",w1,"m/s"
      print*,"shock Mach No.,W11,=",w11
      print*,"shock pressure ratio,P21,=",p21
      print*,"shock overpressure,Ps,=",ps,"kpa"
      print*,"flow velocity ratio,U21,=",u21
      print*,"flow velocity,U2,=",U2,"m/s"
      print*,"sound speed ratio,A21, =",a21
      print*,"sound speed,A2,=",A2,"m/s"
      print*,"temperature ratio,T21,=",t21
      print*,"flow Mach no.,M2,=",m2f
      print*,"density ratio,G21,=",g21
      print*,"dynamic pressure ratio,Q21,=",q21
      print*,"dynamic pressure,Q2,=",Q2,"kPa"
c
      if(M2F.lt.1.)then
c
      print*,"total head pressure ratio,P021,subsonic,=",p021a
      print*,"total head overpressure,P02,=",PstagA,"kPa"
c
      else
c
      print*,"total head pressure ratio,P021,supersonic,=",p021b
      print*,"total head overpressure,P02,=",Pstagb,"kPa"
c
      end if
c
      print*,"reflected temperature ratio,T52,=",t52
      print*,"reflected shock Mach no.,W21,=",w21
```

```
print*,"reflected shock overpressure,Pr,=",dp5,"kpa"
end
```

Appendix D: PTUBE Program Listing and Usage

The PTUBE program can be used to design LB/TS experiments by using it in the backward mode. This is accomplished by specifying the ambient conditions, the throat baffle size, and the desired incident shock overpressure at the test section. The code then uses this input to determine the proper driver gas pressure and temperature to produce this shock. The following example input demonstrates the use of the NAMELIST I/O feature of Fortran 77 to define the input. In this example, English units are defined. Thus, the pressure is in *psi* and the temperature is in degrees Fahrenheit. The baffle size is specified as the percentage of available flow area in the throat. Here, 100% implies that no baffle is used; the full throat area is available for gas to exit the driver. The other valid option for the variable *units* is **SI**.

```
$input
  mode   = 'backward'
  units  = 'english'
  pamb   =  12.32
  tamb   =  60.0
  baffle = 100
  shock  =  12.0
$end
```

The output of this case is listed below. One can see that, regardless of the units specified in the input, the code produces output in both English and SI units. If SI units are specified, then temperature is provided in degrees Celcius and the pressure is in kPa.

```
Results Summary

Throat Baffle Size  =       100 %

Ambient Pressure    =    84.944 kPa
Ambient Pressure    =    12.320 psi

Ambient Temperature =    15.556 deg C
Ambient Temperature =    60.000 deg F

Driver  Pressure    =  3439.355 kPa gauge
Driver  Pressure    =   498.833 psi gauge

Driver  Temperature =   141.349 deg C
Driver  Temperature =   286.428 deg F

Shock   Pressure    =    82.738 kPa gauge
Shock   Pressure    =    12.000 psi gauge

Normal Program Termination
Program Stopped!
```

The next example illustrates use of the PTUBE program in forward mode. Here, the ambient conditions and the driver gas pressure are specified in order to determine the proper driver temperature and expected incident shock overpressure at the test section.

```
$input
  mode   = 'forward'
  units  = 'english'
  pamb   =  12.32
  tamb   =  60.0
  baffle = 100
  pdrv   = 400.0
$end
```

The resulting PTUBE output is as follows.

```
Results Summary

Throat Baffle Size   =      100 %

Ambient Pressure     =   84.944 kPa
Ambient Pressure     =   12.320 psi

Ambient Temperature =   15.556 deg C
Ambient Temperature =   60.000 deg F

Driver  Pressure     = 2757.920 kPa gauge
Driver  Pressure     =  400.000 psi gauge

Driver  Temperature =  119.750 deg C
Driver  Temperature =  247.550 deg F

Shock   Pressure     =   69.389 kPa gauge
Shock   Pressure     =   10.064 psi gauge

Normal Program Termination
Program Stopped!
```

```
      program ptube
c
c     a code for use with the dna large blast/thermal simulator
c     and the arl probative tube
c
c     this program determines combinations of driver pressure,
c     driver temperature, throat baffle size and shock overpressure,
c     based emperically on experimental and computational research.
c
c     the code can be used in two ways:
c
c     1. the user gives the code the desired driver pressure
c        and the baffle size being used, and the code predicts
c        the shoch strength in the expansion tunnel and also
c        determines the required driver temperature for density
c        matching across the contact surface.
c
c     2. the user gives the code the desired shock strength
c        in the expansion tunnel and the baffle size being used,
c        and the code determines the proper driver pressure and
c        temperature to use to produce that environment.
c
      include 'data.h'
c
c     read input data
c
      call readin
c
c     convert enlish units to si
c
      if (units.eq.'e'.or.units.eq.'E') then
        call english
      else
        tamb = tamb + 273.15
      endif
c
c     normalize data
c
      call normal
c
c     find the baffle data to use
c
      call fndbaf
c
c     calculate driver pressure or shock pressure
c
      if (mode.eq.'f'.or.mode.eq.'F') then
        call forwrd
```

```
      elseif (mode.eq.'b'.or.mode.eq.'B') then
        call bakwrd
      else
        write (*,*) 'Invalid value of variable named mode'
        call stopit
      endif
c
c     calculate driver temperature
c
      call temper
c
c     write out results
c
      call write
c
      write (*,*) 'Normal Program Termination'
      call stopit
      end
```

```
      subroutine readin
c
c     this subroutine reads the input data using namelist
c     and checks the validity of the values
c
c     description of input variables
c
c     mode   = 'f' or 'F' for forward calculation
c              predict shock strength given driver pressure
c              (input variable shock is ignored)
c            = 'b' or 'B' for backward calculation
c              predict driver pressure given shock strength
c              (input variable pdrv is ignored)
c
c     units  = 'e' or 'E' for english units
c              's' or 'S' for si units
c
c     pamb   = ambient pressure (in psi or kpa)
c
c     tamb   = ambient temperature (in deg f or deg c)
c
c     baffle = baffle size
c
c     shock  = shock strength (in psi or kpa)
c
c     pdrv   = driver pressure (in psi or kpa)
c
c
c
      include 'data.h'
c
      namelist /input/ mode,units,pamb,tamb,baffle,shock,pdrv
      character fname*20
c
c     open input file
c
      write (*,*) 'Enter the name of the input file.'
      read  (*,'(a20)') fname
      open  (10,file=fname,status='old',err=10)
      goto 20
   10 write (*,*) 'Unable to open that file'
      call stopit
   20 continue
c
c     read namelist input
c
      read (10,nml=input,err=30)
      goto 40
```

```
30 write (*,*) 'Error reading namelist input'
   call stopit
40 continue
c
c     check validity of some input variables
c
      if (mode.ne.'F'.and.mode.ne.'f'.and.
     1    mode.ne.'B'.and.mode.ne.'b') then
        write (*,*) 'Invalid value for variable named mode'
        call stopit
      endif
c
      if (units.ne.'E'.and.units.ne.'e'.and.
     1    units.ne.'S'.and.units.ne.'s') then
        write (*,*) 'Invalid value for variable named units'
        call stopit
      endif
c
      if (pamb.lt.0.0) then
        write (*,*) 'Negative value for variable named pamb.'
        call stopit
      endif
c
      if (tamb.lt.0.0) then
        write (*,*) 'Negative value for variable named tamb.'
        call stopit
      endif
c
      if (shock.lt.0.0) then
        write (*,*) 'Negative value for variable named shock.'
        call stopit
      endif
c
      if (pdrv.lt.0.0) then
        write (*,*) 'Negative value for variable named pdrv.'
        call stopit
      endif
c
      newbaf = 0.0
      do 50 i=1,nbaffs
        if (baffle.eq.baffls(i)) then
          newbaf = baffle
        endif
50 continue
      if (newbaf.ne.baffle) then
        write (*,*) 'The baffle you have specified is not supported.'
        call stopit
      endif
```

```
c
      return
      end
```

```
      subroutine english
c
c     this subroutine converts the input data from english units
c     to si units so the calculation is done consistently
c
      include 'data.h'
c
      pamb  = pamb *6.8948
      shock = shock*6.8948
      pdrv  = pdrv *6.8948
      tamb  =  (tamb-32)/1.8 + 273.15
c
      return
      end
```

```
      subroutine normal
c
c     this subroutine normalizes the input data
c
      include 'data.h'
c
      prat = (pdrv +pamb)/pamb
      srat = (shock+pamb)/pamb
c
      return
      end
```

```
      subroutine fndbaf
c
c     this subroutine checks to see that the baffle specified in the
c     input data is supported by the program
c     if a baffle is found, the arrays dpr and spr are defined
c
      include 'data.h'
      logical found
c
      found = .false.
      do 10 i=1,nbaffs
        if (baffle.eq.baffls(i)) then
          found = .true.
        endif
   10 continue
c
      if (found) then
        if (baffle.eq.100) call assign(npoints,dpr100,spr100,dpr,spr)
        if (baffle.eq. 90) call assign(npoints,dpr090,spr090,dpr,spr)
        if (baffle.eq. 80) call assign(npoints,dpr080,spr080,dpr,spr)
        if (baffle.eq. 70) call assign(npoints,dpr070,spr070,dpr,spr)
        if (baffle.eq. 60) call assign(npoints,dpr060,spr060,dpr,spr)
        if (baffle.eq. 50) call assign(npoints,dpr050,spr050,dpr,spr)
        if (baffle.eq. 40) call assign(npoints,dpr040,spr040,dpr,spr)
      else
        write (*,*) 'Baffle specified in input is not supported!'
        call stopit
      endif
c
      return
      end
```

```
      subroutine assign (n,in1,in2,out1,out2)
c
c     this subroutine assigns the values of two 1-d arrays to two others
c
      real in1,in2
      dimension in1(n),in2(n),out1(n),out2(n)
c
      do 10 i=1,n
        out1(i) = in1(i)
        out2(i) = in2(i)
   10 continue
c
      return
      end
```

```
      subroutine forwrd
c
c     this subroutine calculates the shock strength based on driver pressure
c
      include 'data.h'
c
c     find interval that target driver presure falls in
c
      if (prat.lt.dpr(1)) then
        write (*,*) 'Value of pdrv is too small!'
        call stopit
      endif
c
      if (prat.gt.dpr(npoints)) then
        write (*,*) 'Value of pdrv is too large!'
        call stopit
      endif
c
      do 10 i=2,npoints
        if(prat.gt.dpr(i-1).and.prat.le.dpr(i)) then
          call interp(dpr(i-1),spr(i-1),dpr(i),spr(i),prat,srat)
          return
        endif
   10 continue
c
      write (*,*) 'No interval found in subroutine forwrd!'
      call stopit
      end
```

```
      subroutine bakwrd
c
c     this subroutine calculates the driver pressure based on shock strength
c
      include 'data.h'
c
c     find interval that target shock presure falls in
c
      if (srat.lt.spr(1)) then
        write (*,*) 'Value of shock is too small!'
        call stopit
      endif
c
      if (srat.gt.spr(npoints)) then
        write (*,*) 'Value of shock is too large!'
        call stopit
      endif
c
      do 10 i=2,npoints
        if(srat.gt.spr(i-1).and.srat.le.spr(i)) then
          call interp(spr(i-1),dpr(i-1),spr(i),dpr(i),srat,prat)
          return
        endif
   10 continue
c
      write (*,*) 'No interval found in subroutine forwrd!'
      call stopit
      end
```

```
      subroutine interp (x1,y1,x2,y2,xint,yint)
c
c     this subroutine performs a linear interpolation
c     given independent variables x1 and x2
c     and the associated dependent variables y1 and y2
c     and the input value of xint, the value of yint is found
c     xint must fall between x1 and x2
c
      yint = y1 + (xint-x1)*(y2-y1)/(x2-x1)
c
      return
      end
```

```
      subroutine temper
c
c     this subroutine calculates the driver temperature
c     based on shock strength
c
      include 'data.h'
c
      trat = 0.476053*srat+0.495974
c
      return
      end
```

```
      subroutine write
c
c     this subroutine writes the results to standard output
c
      include 'data.h'
c
c     try to open old file called ptube.out
c     if it exists, delete it and then open
c     new file called ptube.out
c
      open (11,file='ptube.out',status='old',err=60)
      close (11,status='delete')
   60 open (11,file='ptube.out',status='new')
c
      write (11, 5)
c
      write (11,6) baffle
c
      stor = pamb
      write (11,10) stor
      stor = stor/6.8948
      write (11,11) stor
c
      stor = tamb - 273.15
      write (11,20) stor
      stor = 1.8*stor + 32.0
      write (11,21) stor
c
      stor = (prat-1)*pamb
      write (11,30) stor
      stor = stor/6.8948
      write (11,31) stor
c
      stor = trat*tamb - 273.15
      write (11,40) stor
      stor = 1.8*stor + 32.0
      write (11,41) stor
c
      stor = (srat-1)*pamb
      write (11,50) stor
      stor = stor/6.8948
      write (11,51) stor
c
      write (*,*) 'Results written to file ptube.out'
      close (11,status='keep')
c
c     format statements
c
```

```
    5 format (1x,'Results Summary'/)
    6 format (1x,'Throat Baffle Size  =',  i9,' %'/)
   10 format (1x,'Ambient Pressure    =',f9.3,' kPa')
   11 format (1x,'Ambient Pressure    =',f9.3,' psi'/)
   20 format (1x,'Ambient Temperature =',f9.3,' deg C')
   21 format (1x,'Ambient Temperature =',f9.3,' deg F'/)
   30 format (1x,'Driver  Pressure    =',f9.3,' kPa gauge')
   31 format (1x,'Driver  Pressure    =',f9.3,' psi gauge'/)
   40 format (1x,'Driver  Temperature =',f9.3,' deg C')
   41 format (1x,'Driver  Temperature =',f9.3,' deg F'/)
   50 format (1x,'Shock   Pressure    =',f9.3,' kPa gauge')
   51 format (1x,'Shock   Pressure    =',f9.3,' psi gauge'/)
c
      return
      end
```

```
      subroutine stopit
c
c     this subroutine is called to stop the program
c
      write (*,*) 'Program Stopped!'
      stop
      end
```

A listing of the file **data.h** that contains the common blocks and parameter definitions for the PTUBE program.

```
c
      character*1 mode,units
      common /char/   mode,units
      integer baffle
      common /vars/ pamb,tamb,baffle,shock,pdrv,
     1              prat,srat,trat
c
      integer baffls
      parameter (nbaffs  = 7)
      dimension baffls(nbaffs)
      data baffls /40,50,60,70,80,90,100/
c
      parameter (npoints = 6)
c
      dimension dpr040(npoints),spr040(npoints)
      data dpr040 /1.000, 31.500, 69.000,124.500,180.000,232.500/
      data spr040 /1.000,  1.568,  1.902,  2.269,  2.635,  2.954/
c
      dimension dpr050(npoints),spr050(npoints)
      data dpr050 /1.000, 30.000, 66.750,120.250,173.750,226.250/
      data spr050 /1.000,  1.582,  1.948,  2.351,  2.740,  3.098/
c
      dimension dpr060(npoints),spr060(npoints)
      data dpr060 /1.000, 28.500, 64.500,116.000,167.500,220.000/
      data spr060 /1.000,  1.596,  1.993,  2.433,  2.846,  3.241/
c
      dimension dpr070(npoints),spr070(npoints)
      data dpr070 /1.000, 27.000, 62.250,111.750,161.250,213.750/
      data spr070 /1.000,  1.610,  2.039,  2.516,  2.951,  3.385/
c
      dimension dpr080(npoints),spr080(npoints)
      data dpr080 /1.000, 25.500, 60.000,107.500,155.000,207.500/
      data spr080 /1.000,  1.623,  2.085,  2.598,  3.056,  3.529/
c
      dimension dpr090(npoints),spr090(npoints)
      data dpr090 /1.000, 24.000, 57.750,103.250,148.750,201.250/
      data spr090 /1.000,  1.637,  2.130,  2.680,  3.162,  3.672/
c
      dimension dpr100(npoints),spr100(npoints)
      data dpr100 /1.000, 25.000, 51.800, 99.000,142.500,195.000/
      data spr100 /1.000,  1.651,  2.176,  2.762,  3.267,  3.816/
c
      common /arrays/ dpr(npoints),spr(npoints)
c
```

Appendix E: LBTSRWE Program Listing

```
      program rwe
c
c    this program calculates the correct area ratio between the
c    expansion section of a shock tube or blast simulator and the open
c    area of a rarefaction wave eliminator located at its
c    downstream end. in the calculations the rwe is assumed to be
c    a simple converging nozzle open to the atmosphere at the
c    downstream end. the flow behind the shock is assumed to be a
c    one-dimensional, steady, isentropic flow of a perfect gas.
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
c     set basic gas parameters for calculations
c
      call setup
c
c     generate angle, actor, aend and area arrays from end vent geometry
c
      call endvnt
c
c     determine type on input file to be used
c     call appropriate subroutine based on response
c
c     1 = pressure history from q1d station file
c     2 = pressure history from sharc replot file
c
   10 write (*,*) ' Enter type of input file being used:'
      write (*,*) ' 1 = pressure history from q1d station file,'
      write (*,*) ' 2 = pressure history from sharc replot file,'
      write (*,*) ' or any other integer to stop program'
      read  (*,*,err=10) input
c
      if (input.eq.1) then
        call q1d
      elseif (input.eq.2) then
        call sharc
      else
       stop
      endif
c
c     generate rwe area ratio history from flow data
c
      call flow
c
c     generate the end vent control function history
c
```

```
      call func
c
c     place limits on end vent control function
c
      call limits
c
c     create a linear closing function
c
      call linear
c
c     write results to output files
c
      if (dbugon) call debug
      call write
c
      stop
      end
```

```
      subroutine setup
c
      include 'const.h'
c
c     set basic gas parameters for calculations
c     treat air as an ideal gas with constant gamma
c
c     set ambient temperature and pressure equal to
c     white sands conditions as defined by dna
c     pressure in pascals
c     temperature in kelvins
c
      data pi/3.1415927/
      data gamma/1.4000/,rair/287.04/,pt/84944.0/,tamb/288.71/
      alpha = (gamma+1)/(gamma-1)
      delta = (gamma/(gamma-1.0))
      gm = (gamma-1.0)/2.0
c
c     calculate ambient sound speed (a1) in meters per second
c
      a1 = sqrt(gamma*rair*tamb)
c
      return
      end
```

```
      subroutine endvnt
c
c     this subprogram defines the relationship between the actuator position
c     louver angle and open area for the end vend of the LB/TS RWE
c
c     description of variables
c
c     nend  = number of data point in end vent calibration
c     atunl = area of expansion tunnel (square meters)
c     actor = actuator displacement (inches)
c     angle = louver angle (from horizontal in degrees)
c     aend  = end vent open area (square meters)
c     area  = end vent open area ratio
c     lvdt  = end vent lvdt signal (volts)
c
      include 'areas.h'
c
c     define expansion tunnel cross-sectional area
c
      data atunl/165.0/
c
c     define lvdt voltage for calibration
c
      data  lvdt/ 0.671, 1.020, 1.462, 1.841, 2.210, 2.590,
     1            2.960, 3.450, 3.840, 4.260, 4.690, 5.200,
     2            5.700, 6.280, 6.970, 7.860, 8.930,10.000/
c
c     define actuator displacement for calibration
c
      data actor/ 1.205854, 1.833042, 2.627360,
     1            3.308461, 3.971591, 4.654489,
     2            5.319416, 6.199995, 6.900864,
     3            7.655646, 8.428399, 9.344920,
     4           10.243470,11.285788,12.525787,
     5           14.125206,16.048103,17.971000/
c
c     define end vent louver angle for calibration
c
      data angle/ 0, 2, 4, 6, 8,10,12,14,16,
     1           18,20,22,24,26,28,30,32,34/
c
c     define end vent open area for calibration
c
      data aend/ 139.260,130.569,121.888,
     1           113.228,104.600, 96.014,
     2            87.481, 79.011, 70.614,
     3            62.301, 54.082, 45.966,
     4            37.964, 30.085, 22.340,
```

```
     5             14.736,  7.285,  0.000/
c
c     calculate area ratio for end vent
c
      do 10 i=1,nend
        area(i) = aend(i)/atunl
   10 continue
c
      return
      end
```

```
      subroutine q1d
c
c     this subroutine reads brl-q1d station data from
c     the file specified by the user, calculates
c     local sound speed and local mach number
c     and stores the values in the array called val
c
      include 'const.h'
      include 'hist.h'
c
      character junk*132,q1dfile*15
c
c     get filename from user and open the file
c
      write (*,*) ' Enter the name of the brl-q1d station file:'
      read  (*,'(a15)') q1dfile
      open (10,file=q1dfile,status='old',err=10)
c
c     skip header information at top of file
c
c      write (*,*) ' Information found at top of brl-q1d station file'
      do 20 i=1,7
        read  (10,'(a132)') junk
c         write  (*,'(a132)') junk
   20 continue
c
c     read the station data
c
      iter = 1
      isat = 0
      sat  = 0.0
      pmax = 0.0
   30 read (10,*,end=40) val(iter,1),val(iter,3),rho,temp,val(iter,8),
     +                   val(iter,2),val(iter,4),val(iter,7),crit
c
c     find maximum static overpressure and assume it is
c     the shock front. store the peak into pmax,
c     the shock arrival time into sat
c     and the index into isat
c
      if (val(iter,3).gt.pmax) then
        pmax = val(iter,3)
        sat  = val(iter,1)
        isat = iter
      endif
c
c     convert static overpressure to absolute static pressure
c
```

```
      val(iter,3) = val(iter,3)*1000 + pt
c
c     convert absolute stagnation pressure to stagnation overpressure
c
      val(iter,2) = val(iter,2) - pt/1000
c
c     calculate local mach number
c
      val(iter,5) = val(iter,8)/val(iter,7)
c
c     increment counter and test to see that
c     iter does not exceed maxiter
c
      iter = iter + 1
      if (iter.le.maxiter) then
        goto 30
      else
        write (*,*) ' the number of points in the station data file'
        write (*,*) ' is greater than the parameter maxiter.'
        write (*,*) ' increase maxiter and recompile the code.'
        stop
      endif
   40 continue
      iter = iter -1
c      write (*,*) ' '
c      write (*,*) ' Number of records in brl-q1d station data =',iter
c
c       write (*,*) ' isat=',isat
c       write (*,*) '  sat=', sat
c       write (*,*) ' pmax=',pmax
c
      close (10,status='keep')
      return
c
   10 write (*,*) ' error opening brl-q1d station file'
      stop
      end
```

```
      subroutine sharc
c
c     this subroutine reads sharc station data from the file named
c     replot, calculates stagnation pressure, mach number,
c     local sound speed and dynamic pressure and stores the
c     values in the array called val
c
      include 'const.h'
      include 'hist.h'
      dimension rho(maxiter),u(maxiter),v(maxiter)
c      character junk*72
c
c     open sharc station data file 'replot'
c
      open (10,file='replot',status='old',err=10)
c
c     read header information
c
      read (10,*) iter,nsta,prob,xsta,ysta
c      read (10,'(a72)') junk
c      read (10,'(a72)') junk
c
c     test to be sure iter is not greater than maxiter
c
      if (iter.gt.maxiter) then
        write (*,*) ' the number of points in the history file'
        write (*,*) ' is greater than the parameter maxiter.'
        write (*,*) ' increase maxiter and recompile the code.'
        stop
      endif
c
c     read data from replot file, find shock arrival time
c     convert to si units and prepare data for area subroutines
c
      isat = 0
      sat  = 0.0
      pmax = 0.0
      do 20 i=1,iter
        read (10,*) val(i,1),val(i,3),rho(i),u(i),v(i)
c
c     find maximum static overpressure and assume it is
c     the shock front. store the peak into pmax,
c     the shock arrival time into sat
c     and the index into isat
c
      if (val(i,3).gt.pmax) then
        pmax = val(i,3)
        sat  = val(i,1)
```

```
        isat = i
      endif
c
c       convert units from replot file into si units
c
c       from overpressure in dynes per square centimeters
c       to absolute pressure in pascals
c
        val(i,3) = val(i,3)/10.0 + pt
c
c       density in grams per cubic centimeter
c               to kilograms per cubic meter
c
        rho(i) = rho(i) * 1000.0
c
c       velocity in centimeters per second
c                to meters per second
c
        u(i) = u(i)/100.0
        v(i) = v(i)/100.0
c
c       calculate absolute velocity, sound speed and local mach number
c
        vsqrd = u(i)**2+v(i)**2
        csqrd = gamma*val(i,3)/rho(i)
        sqm = vsqrd/csqrd
c        if(sqm.lt.1.0e-02) sqm=0.0
        val(i,5) = sqrt(sqm)
        val(i,7) = sqrt(csqrd)
        val(i,8) = sqrt(vsqrd)
c
c       calculate dynamic pressure in kilopascals
c
        val(i,4) = gamma*val(i,3)*sqm/2000.0
c
c       calculate stagnation overpressure in kilopascals
c
        val(i,2) = val(i,3)*((1+gm*sqm)**delta)
        val(i,2) = (val(i,2) - pt)/1000.0
c
   20 continue
      close (10,status='keep')
      return
c
   10 write (*,*) ' error opening sharc history file named replot'
      stop
      end
```

```
      subroutine flow
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
c     generate array of rwe open area from input data
c     find max rwe open area in the array (aramax)
c     hold rwe open area to the setting for shock arrival
c     for all times prior to shock arrival
c
c     this flag determines which area ratio subroutine is used
c
      method = 2
c
      if (method.eq.1) then
        call area1 (pt,a1,val(isat,1),val(isat,3),
     1                    val(isat,7),val(isat,8),aramax)
      endif
c
      if (method.eq.2) then
          ps1 = val(isat,2)*1000+pt
          call area2 (ps1,val(isat,5),aramax)
      endif
c
      do 10 i=1,isat
        val(i,3)=(val(i,3)-pt)/1000.0
        val(i,6)=aramax
   10 continue
c
      do 20 i=isat+1,iter
c
        if (method.eq.1) then
          call area1 (pt,a1,val(i,1),val(i,3),val(i,7),val(i,8),ratio)
        endif
c
        if (method.eq.2) then
          ps1 = val(isat,2)*1000+pt
          call area2 (ps1,val(i,5),ratio)
        endif
c
        val(i,3)=(val(i,3)-pt)/1000.0
        val(i,6)=ratio
        if (ratio.gt.aramax) aramax = ratio
   20 continue
c
      return
      end
```

```
      subroutine area1 (p1,a1,t,p2,a2,u2,ratio)
c
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccc                                                              ccc
ccc       reflection eliminator program by james gottlieb        ccc
ccc                                                              ccc
ccc     for the denver research institute  (4 january 1987)      ccc
ccc                                                              ccc
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c-------------------------------------------------------------------
c       this is the main computer program to compute the area setting
c  for the reflection eliminator to produce no reflected wave except
c  for the transient spike, or to obtain the reflected wave strength
c  when the area setting of the reflection eliminator is specified.
c  the four special cases under consideration are summarized below:
c
c    1) flat-topped incident shock wave with a given pressure ratio,
c       specified in the input data file called in, which results in
c       no wave reflection except for the transient spike (note that
c       only one value of the final area setting will be returned to
c       the user in the output data file called out for each input),
c
c    2) same case of a flat-topped incident shock wave as in case a,
c       but this time it is is repeated for the case of a reflected
c       wave with a given pressure ratio in terms of a percentage of
c       the incident shock pressure ratio (greater or less than zero
c       for a reflected shock or rarefaction wave, respectively),
c
c    3) flat-topped incident shock wave with a given pressure ratio
c       and a specified area setting for the reflection eliminator,
c       which are given in the input data file called in, for which
c       a reflected shock or rarefaction wave will in general occur
c       and the strength of this reflected wave will be computed and
c       put in the output data file called out,
c
c    4) incident blast wave with a time varying signature for which
c       the pressure, sound speed, flow velocity and gamma are given
c       as a function of time in the input file called in, which now
c       produces no reflected wave (the same number of values of the
c       output as input will be returned to the user in the output
c       data file called out).
c
c  for each type of reflection eliminator run the first line in the
c  input data file in must have the initial or atmospheric pressure
c  and sound speed, in units of pa (n/m2) and m/s.
c
c-------------------------------------------------------------------
      real     ms,  m3,  mj,  mflux
```

```
      data      pi/3.141592654/
      radin = 0.0
      radout= 0.0
      width = 0.254
      g = 1.400
c-----------------------------------------------------------------------
c
c  compute the jump in flow properties across the reflected shock or
c  rarefaction wave to state 3, if a reflected wave actually exists.
c
      numrw=0
   30 numrw=numrw+1
      p3=p2
      a3=a2
      u3=u2
      m3=u3/a3
c-----------------------------------------------------------------------
c
c  for outflow (from the channel through the reflection eliminator),
c  m3 and p3-p1 are both positive, and the following coding is used.
c
      if (m3.le.0.0 .or. p3-p1.le.0.0) goto 50
      pcrit=p3*((2.0+(g-1.0)*m3*m3)/(g+1.0))**(g/(g-1.0))
      if (pcrit .ge. p1) then
          pj=pcrit
          mj=1.0
      else
          pj=p1
          mj=sqrt(((pcrit/pj)**((g-1.0)/g)*(g+1.0)-2.0)/(g-1.0))
      endif
          aj=a3*(pj/p3)**((g-1.0)/(2.0*g))
          uj=mj*aj
          areaj=(m3/mj)*(p3/pj)**((g+1.0)/(2.0*g))
      tau3=(g-1.0)*m3*m3/(2.0+(g-1.0)*m3*m3)
      tauj=(g-1.0)*mj*mj/(2.0+(g-1.0)*mj*mj)
      zeta=tauj/(1.0-tauj)
      c0=pi/(pi+2.0-5.0*zeta+2.0*zeta*zeta)
      eta=7.0*tauj+1.0/(1.0+12*tauj)
      a=(2.0*eta-1.0)*(1.0-c0)
      b=2.0*(1.0-eta)*(1.0-c0)
      cds=c0+a*(tau3/tauj)+b*(tau3/tauj)**2
              cd=cds
              numcd=1
   40 z=-45.0*radout*cd/(areaj*width)
      if (abs(z) .gt. 78) z=78.0*z/abs(z)
      omega=1.0-exp(z)
      cdprev=cd
      cd=cds+omega*(1.0-cds)
```

```
      if (abs(1.0-cdprev/cd) .lt. 5.0e-05) goto 100
          numcd=numcd+1
          if (numcd .lt. 20) goto 40
          write (*,*) ' failure of cd iteration in subroutine area1'
          stop
c---------------------------------------------------------------------
c
c  for inflow (through the reflection eliminator to the channel), m3
c  and p3-p1 are both negative and the following coding is employed.
c
   50 if (m3.ge.0.0 .or. p3-p1.ge.0.0) goto 90
      gg=(g-1.0)/2.0
      a3=sqrt(a1*a1-gg*u3*u3)
      mflux=g*p3*u3/(a3*a3)
      z=((g+1.0)/2.0)**((g+1.0)/(2.0*g-2.0))*mflux*a1/(g*p1)
      pcrit=p1*(2.0/(g+1.0))**(g/(g-1.0))*(1.0-(1.0-z)**2*g/2.0)
      if (pcrit .ge. p3) then
          mj=-1.0
      else
          kcycl=0
          mj=-sqrt((p1-p3)/(p1-pcrit))
          z=mflux*a1/(g*p1)
   60     zz=mj-z*(1.0+gg*mj*mj)**((g+1.0)/(2.0*g-2.0))
          ff=p3*(1.0+gg*mj*mj)**(g/(g-1.0))-p1*(1.0-g*zz*zz/2.0)
          fp=g*p3*mj*(1.0+gg*mj*mj)**(1.0/(g-1.0))+g*p1*zz*(1.0-(g
     +        +1.0)*(z/2.0)*mj*(1.0+gg*mj*mj)**((3.0-g)/(2.0*g-2.0)))
          prev=mj
          mj=mj-ff/fp
          kcycl=kcycl+1
          if (abs(mj-prev) .lt. 0.001) goto 70
          if (kcycl .lt. 30) goto 60
          write (*,*) ' failure of mj iteration in subroutine area1'
          stop
      endif
   70 areaj=(z/mj)*(1.0+gg*mj*mj)**((g+1.0)/(2.0*g-2.0))
      cds=0.5+mj**2/8.0+(2.0-g)*mj**4/48.0
     +        +(2.0-g)*(3.0-2.0*g)*mj**6/384.0
          cd=cds
          numcd=1
   80 z=-45.0*radin*cd/(areaj*width)
      if (abs(z) .gt. 78) z=78.0*z/abs(z)
      omega=1.0-exp(z)
      cdprev=cd
      cd=cds+omega*(1.0-cds)
      if (abs(1.0-cdprev/cd) .lt. 5.0e-05) goto 100
          numcd=numcd+1
          if (numcd .lt. 20) goto 80
          write (*,*) ' failure of cd iteration in subroutine area1'
```

```
          stop
c-------------------------------------------------------------------
c
c  in the two cases when p3-p1 and m3 have the opposite signs, there
c  is no reflection elimination, and the reflection eliminator area
c  is simply set to zero.
c
   90 areaj=0.0
      cd=1.0
c-------------------------------------------------------------------
c
  100 areae=areaj/cd
      if (areaj .lt. 0.0) areaj=0.0
      if (areaj .gt. 1.0) areaj=1.0
      if (areae .lt. 0.0) areae=0.0
      if (areae .gt. 1.0) areae=1.0
c
c     until we figure out who cd relates to
c     our rwe we will use areaj as the rwe area. (sjs)
      ratio=areaj
c
      return
      end
```

```
      subroutine area2 (ps1,zmach,ratio)
c
c ***********************************************************************
c
c        this subroutine calculates the rwe open area as a function of the
c     ambient pressure, stagnation pressure at the input to the rwe and the
c     mach number at the input of the rwe
c
c     pt         = ambient pressure
c     ps1        = stagnation pressure at the input to the rwe
c     ps1t       = ps1/pt
c     zmach      = mach number at the input to the rwe
c     mt         = mach number at the rwe exit plane
c     ratio      = required rwe open area ratio
c
      real mt
      include 'const.h'
c
c    calculation of the isentropic mach number at the downsteam
c    end of the rwe (mt)
c
      ps1t = ps1/pt
      if (ps1t.lt.1.0) then
        ratio=0.
        return
      endif
      if (ps1t.ge.1.892936) then
        mt = 1.0
      else
        mt = ((2.0/(gamma-1.0))*(ps1t**(1/delta)-1.0))**0.5
      endif
c
c    calculation of rwe area ratio (ratio)
c
      ratio = (1.0/mt)*((2.0+(gamma-1.0)*mt**2.0)/2.0)**(alpha/2)*
     +(zmach)*((2.0+(gamma-1.0)*zmach**2.0)/2.0)**(-1.0*alpha/2)
c
      return
      end
```

```
      subroutine func
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
c     calculate:
c       1. end vent actuator displacement (inches)
c       2. end vent louver angle (degrees)
c     from rwe open area history by interpolation
c
c     this routine assumes side vents are passive and subtracts off residual
c     open area of side vent when calculating end vent louver angle
c
      if (aramax.gt.area(1)) then
        aside = aramax-area(1)
        if (dbugon) then
          write (*,*) ' Maximum required rwe area greater'
          write (*,*) ' than available end vent area.'
          write (*,*) ' Required side vent area ratio=',aside
          write (*,*) ' Required side vent area      =',aside*atunl
        endif
      else
        aside = 0.0
      endif
c
      do 20 i=1,iter
c
c       calculate required end vent area as the difference between
c       the required open area and the side vent area
c       if end vent area falls below zero, set equal to zero
c
        val(i,17) = val(i, 6)
        val(i, 6) = val(i,17) - aside
        if (val(i,6).lt.0.0) val(i,6) = 0.0
c
c       find end vent louver angle, actuator piston displacement
c       and lvdt voltage for this end vent area ratio
c
        call lookup1 (val(i,6),val(i,9),val(i,10),val(i,19))
c
   20 continue
c
c     calculate:
c       1. end vent louver angular velocity (radians per second)
c       2. end vent actuator piston velocity (inches per second)
c
      do 30 i=1,iter
```

```
        if (i.eq.1) then
          val(i,11) = (pi/180.0)*(val(i+1,10) - val(i  ,10)) /
     1                           (val(i+1, 1) - val(i  , 1))
          val(i,12) =            (val(i+1, 9) - val(i  , 9)) /
     1                           (val(i+1, 1) - val(i  , 1))
        elseif (i.eq.iter) then
          val(i,11) = (pi/180.0)*(val(i  ,10) - val(i-1,10)) /
     1                           (val(i  , 1) - val(i-1, 1))
          val(i,12) =            (val(i  , 9) - val(i-1, 9)) /
     1                           (val(i  , 1) - val(i-1, 1))
        else
          val(i,11) = (pi/180.0)*(val(i+1,10) - val(i-1,10)) /
     1                           (val(i+1, 1) - val(i-1, 1))
          val(i,12) =            (val(i+1, 9) - val(i-1, 9)) /
     1                           (val(i+1, 1) - val(i-1, 1))
        endif
   30 continue
c
      return
      end
```

```
      subroutine lookup1 (a,b,c,d)
c
c     this subroutine finds the actuator position, louver angle
c     and lvdt voltage to generate a given rwe open area ratio.
c     the first value, a, in the subroutine statement is the required
c     open area ratio. the last three values are the actuator displacement,
c     louver angle and lvdt voltage respectively, which are returned values.
c
      include 'areas.h'
c
      if (a.ge.area(1)) then
        b = actor(1)
        c = angle(1)
        d = lvdt(1)
        return
      endif
c
      do 10 i=2,nend
        if (a.le.area(i-1).and.a.gt.area(i)) goto 20
        if (a.eq.area(i)) then
          b = actor(i)
          c = angle(i)
          d = lvdt(i)
          return
        endif
   10 continue
   20 continue
c
      d =  lvdt(i-1) + (a       -area (i-1)) *
     +                 ( lvdt(i)- lvdt(i-1)) /
     +                 (area (i)-area (i-1))
c
      c = angle(i-1) + (a       -area (i-1)) *
     +                 (angle(i)-angle(i-1)) /
     +                 (area (i)-area (i-1))
c
      b = actor(i-1) + (a       -area (i-1)) *
     +                 (actor(i)-actor(i-1)) /
     +                 (area (i)-area (i-1))
c
      return
      end
```

```
      subroutine limits
c
c     this subroutine limits the motion, velocity and acceleration
c     of the final rwe control function
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
      real mult
      real knee
      data knee/3.0/,vellow/27.8/,velhi/83.4/,accel/2000/
c
c     eliminate reversals in direction
c
c     this section of code searches the end vent actuator displacement
c     history and searches for hill-valley pairs and then creates
c     a new actuator function with no reversals in direction
c
c     copy unlimited actuator piston displacement array values
c     into limited actuator piston displacement array without modification
c
      do 10 i=1,iter
        val(i,13) = val(i,9)
   10 continue
c
c     find a hill-valley pair
c
   20 do 30 i=2,iter
        npairs = 0
        if (val(i,13).lt.val(i-1,13)) then
          npairs = npairs + 1
          ihill = i-1
          do 40 j=ihill+1,iter-1
            if (val(j,13).gt.val(j-1,13)) then
              ival = j -1
              goto 50
            endif
   40     continue
          goto 75
        endif
   30 continue
   50 continue
c
c     calculate a plateau value and
c     smooth through this pair
c
      if (npairs.ne.0) then
```

```
c
        if (dbugon) then
          write (*,*) ' pair found at:'
          write (*,*) ' ihill =',ihill
          write (*,*) ' ival  =',ival
          write (*,*) ' '
        endif
c
        plateau = (val(ihill,13) + val(ival,13)) * 0.5
c
c     find first point before ihill which exceeds plateau
c
        do 60 i=1,ihill
          if (val(i,13).ge.plateau) then
            iplat1 = i
            goto 61
          endif
   60   continue
   61   continue
c
c     find first point after ival which exceeds plateau
c
        do 62 i=ival,iter
          if (val(i,13).ge.plateau) then
            iplat2 = i
            goto 63
          endif
   62   continue
   63   continue
c
c     set all values between iplat1 and iplat2 to plateau
c
        do 64 i=iplat1,iplat2
          val(i,13) = plateau
   64   continue
c
c     go back and look for another pair
c
   70   goto 20
      endif
   75 continue
c
c     limit actuator piston velocity and acceleration
c
c     this section of code limits the actuator piston velocity to
c     27.8 in/s for displacements which are < 3 inches and
c     83.4 in/s for displacements which are > 3 inches
c
```

```
c     it also limits the actuator piston acceleration/deceleration to
c     2000 in/s**2 for the entire stroke
c
      do 80 i=1,iter
c
c     calculate new piston velocity history based on reversal limitation
c
        if (i.eq.1) then
          val(i,14) =           (val(i+1,13) - val(i  ,13)) /
     1                          (val(i+1, 1) - val(i  , 1))
        elseif (i.eq.iter) then
          val(i,14) =           (val(i  ,13) - val(i-1,13)) /
     1                          (val(i  , 1) - val(i-1, 1))
        else
          val(i,14) =           (val(i+1,13) - val(i-1,13)) /
     1                          (val(i+1, 1) - val(i-1, 1))
        endif
c
c     find velocities which exceed limits
c
        if (val(i,13).lt.knee.and.val(i,14).gt.vellow) then
          val(i,14) = vellow
        elseif (val(i,13).ge.knee.and.val(i,14).gt.velhi) then
          val(i,14) = velhi
        endif
c
c     find changes in velocity which exceed acceleration limit
c
        if (i.eq.1) then
          val(i,15) = 0.0
        else
          val(i,15) = (val(i  ,14) - val(i-1,14)) /
     1                (val(i  , 1) - val(i-1, 1))
          if (abs(val(i,15)).gt.accel) then
            if (val(i,15).lt.0.0) then
              mult = -1.0
            else
              mult =  1.0
            endif
            val(i,15) = mult * accel
            val(i,14) = val(i-1,14) + val(i,15)*(val(i,1) - val(i-1,1))
            if (val(i,14).lt.0.0) val(i,14) = 0.0
          endif
        endif
c
c     reconstruct piston displacement history
c     based on limited velocity and acceleration
c
```

```
          if (i.eq.1) then
            val(i,13) = val(i,9)
          elseif (i.eq.iter) then
            t2 = val(i  ,1)
            t1 = val(i-1,1)
            val(i,13) = val(i-1,13) + val(i,14)*(t2-t1)
            if (val(i,13).lt.val(i-1,13)) val(i,13) = val(i-1,13)
          else
            t2 = (val(i+1,1) + val(i  ,1)) * 0.5
            t1 = (val(i  ,1) + val(i-1,1)) * 0.5
            val(i,13) = val(i-1,13) + val(i,14)*(t2-t1)
          endif
c
c     calculate area ratio and lvdt voltage
c     for rwe function with restrictions
c
          call lookup2 (val(i,13),val(i,16),val(i,20))
c
c     build total area ratio function
c
          val(i,18) = val(i,16) + aside
c
   80 continue
c
      return
      end
```

```
      subroutine linear
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
c     this subroutine generates a linear rwe closing function
c     the rwe open area is held constant until shock arrival,
c     closes linearly between shock arrival and the minimum
c     open area and then holds constant once reaching the minimum
c
c     loop through restricted actuator position history
c     and find point of maximum extension (fully closed)
c
      actmax = 0.0
      do 10 i=1,iter
        if (val(i,13).gt.actmax) then
          ippd = i
          actmax = val(ippd,13)*1.0001
        endif
   10 continue
      aramin = val(ippd,18)
c
      if (dbugon) then
        write (*,*) ' rwe reached minimum opening at:'
        write (*,*) ' time (s) = ',val(ippd, 1)
        write (*,*) ' area (%) = ',aramin
      endif
c
c     loop through restricted actuator position history again
c     if time is before shock arrival, hold constant at max open
c     if time is after min opening, hold constant at min open
c     if time is between shock arrival and
c
      do 20 i=1,iter
        if (i.lt.isat.or.i.gt.ippd) then
          val(i,21) = val(i,13)
          val(i,22) = val(i,16)
          val(i,23) = val(i,18)
          val(i,24) = val(i,20)
        else
          val(i,21) = val(isat,13) + (val(i    , 1) - val(isat, 1))
     1                             * (val(ippd,13) - val(isat,13))
     2                             / (val(ippd, 1) - val(isat, 1))
          call lookup2(val(i,21),val(i,22),val(i,24))
          val(i,23) = val(i,22) + aside
        endif
   20 continue
```

```
c
      return
      end
```

```
      subroutine lookup2 (a,b,c)
c
c     this subroutine finds the rwe open area and lvdt voltage
c     given an actuator piston position. the first value, a,
c     in the subroutine statement is the input piston
c     position, the second value, b, is the returned
c     rwe end vent area ratio and the third value, c,
c     is the returned restricted lvdt voltage signal
c
      include 'areas.h'
c
      if (a.eq.actor(1)) then
        b = area(1)
        c = lvdt(1)
        return
      endif
c
      do 10 i=2,nend
        if (a.gt.actor(i-1).and.a.le.actor(i)) goto 20
        if (a.eq.actor(i)) then
          b = area(i)
          c = lvdt(i)
          return
        endif
   10 continue
   20 continue
c
      b = area (i-1) + (a       -actor(i-1)) *
     +                 (area (i)-area (i-1)) /
     +                 (actor(i)-actor(i-1))
c
      c = lvdt (i-1) + (a       -actor(i-1)) *
     +                 (lvdt (i)-lvdt (i-1)) /
     +                 (actor(i)-actor(i-1))
c
      return
      end
```

```
      subroutine debug
c
c     this subroutine writes the vectors 1 though maxvals
c     to separate output files for debugging purposes
c
c     for normal use of the code, output produced by
c     the write subroutine is used
c
      include 'hist.h'
      character filename*6
c
      do 10 j=1,maxvals
        write (filename,'(a4,i2)') 'rwe.',j+10
        open (11,file=filename,status='new',err=40)
        do 20 i=1,iter
          write (11,30) val(i,j)
   20   continue
        close (11,status='keep')
   10 continue
c
   30 format (1x,e13.5,1x,',')
c
      return
c
c     stop program if rwe output file already exists
c
   40 write (*,*) ' '
      write (*,*) ' Unable to open new file ',filename
      write (*,*) ' Delete rwe output files and run program again.'
      write (*,*) ' Program stopped.'
      stop
c
      end
```

```
      subroutine write
c
      include 'areas.h'
      include 'const.h'
      include 'hist.h'
c
      parameter (nfile = 4)
      character*8 outname(nfile)
      data outname/ 'rwe.pres','rwe.area','rwe.disp','rwe.lvdt'/
c
c     this subroutine write the results to the output file rweout.dat
c
c     open output files
c
      do 5 i=1,nfile
        open (10+i,file=outname(i),status='new',err=30)
    5 continue
c
c     write header information at top of file
c
      do 6 i=1,nfile
       write(10+i,50)pt/1000.0,tamb-273.15,sat,pmax,aramax*100,aside*100
    6 continue
c
c     write column headers
c
      write (11,41)
      write (12,42)
      write (13,43)
      write (14,44)
c
c     write history data
c
      do 10 i=1,iter
        write (11,21) val(i,1),val(i, 3),val(i, 2),val(i,4)
        write (12,22) val(i,1),val(i,18)*100,val(i,23)*100
        write (13,22) val(i,1),val(i,13),val(i,21)
        write (14,22) val(i,1),val(i,20),val(i,24)
   10 continue
c
      return
c
c     format statements
c
   21 format (4(1x,e11.4))
   22 format (3(1x,e11.4))
c
   50 format (2x,'SUMMARY OF RWE OUTPUT DATA'//
```

```
     1         2x,'Ambient Pressure (kPa) =',e11.4/
     2         2x,'Ambient Temperature (C) =',e11.4/
     3         2x,'Time of Shock Arrival (s) =',e11.4/
     4         2x,'Shock Overpressure (kPa) =',e11.4/
     5         2x,'Maximum RWE Open Area (%) =',e11.4/
     6         2x,'Required Side Vent Open Area (%) =',e11.4/)
c
   41 format ( 2x,'RWE PRESSURE DATA'//
     1         14x,'Static',6x,'Stagnation',2x,'Dynamic'/
     2          2x,'Time',8x,'Overpress',3x,'Overpress',3x,'Pressure'/
     3          2x,'(s)',9x,'(kPa)',7x,'(kPa)',7x,'(kPa)'/)
c
   42 format ( 2x,'RWE AREA DATA'//
     1         14x,'RWE',9x,'Linear RWE'/
     2          2x,'Time',8x,'Open Area',3x,'Open Area'/
     3          2x,'(s)',9x,'(%)',9x,'(%)'/)
c
   43 format ( 2x,'RWE ACTUATOR DATA'//
     1         26x,'Linear'/
     2         14x,'Actuator',4x,'Actuator'/
     3          2x,'Time',8x,'Position',4x,'Position'/
     4          2x,'(s)',9x,'(in)',8x,'(in)'/)
c
   44 format ( 2x,'RWE LVDT DATA'//
     1         26x,'Linear'/
     2         14x,'LVDT',8x,'LVDT'/
     3          2x,'Time',8x,'Signal',6x,'Signal'/
     4          2x,'(s)',9x,'(volts)',5x,'(volts)'/)
c
c     stop program if rwe output file already exists
c
   30 write (*,*) ' '
      write (*,*) ' Unable to open new file ',outname(i)
      write (*,*) ' Delete rwe output files and run program again.'
      write (*,*) ' Program stopped.'
      stop
c
      end
```

A listing of the file **areas.h** that contains the common block and parameter definition to allocate storage for the end vent data lookup table for the PTUBE program.

```
      parameter (nend = 18)
      real lvdt
      common /areas/ actor(nend),angle(nend),aend(nend),
     1               area(nend), lvdt(nend),
     2               aramax,atunl,aside,aramin,ippd
```

A listing of the file **const.h** that contains common block definitions to allocate storage for constants used by the PTUBE program.

```
      common/const/gamma,alpha,delta,gm,rair,pt,tamb,a1,pi
      logical dbugon
c
c     only one of the following two lines may be used
c     comment out the one which is not wanted
c
      data dbugon/.true./
c      data dbugon/.false./
```

A listing of the file **hist.h** that contains common block and paramter definitions to allocate storage for the RWE history data in the PTUBE program.

```
      parameter (maxiter=5000)
      parameter (maxvals=  24)
      common/hist/iter,val(maxiter,maxvals),isat,sat,pmax
c
c     the array val is assigned the following parameters:
c       val(i,1) = time (s)
c       val(i,2) = stagnation overpressure (kPa)
c       val(i,3) = static overpressure (kPa)
c       val(i,4) = dynamic pressure (kPa)
c       val(i,5) = mach number
c       val(i,6) = end vent open area ratio [no restrictions]
c       val(i,7) = local sound speed (m/s)
c       val(i,8) = flow velocity (m/s)
c       val(i,9) = actuator displacement (in) [no restrictions]
c       val(i,10)= louver angle (deg) [no restrictions]
c       val(i,11)= louver angular velocity (rad/sec) [no restrictions]
c       val(i,12)= actuator piston velocity (in/sec) [no restrictions]
c       val(i,13)= actuator displacement (in) [restricted]
c       val(i,14)= actuator piston velocity (in/sec) [restricted]
c       val(i,15)= actuator piston acceleration (in/sec**2) [restricted]
c       val(i,16)= end vent open area ratio [restricted]
c       val(i,17)= rwe open area ratio (end and side vents) [no restrictions]
c       val(i,18)= rwe open area ratio (end and side vents) [restricted]
c       val(i,19)= end vent lvdt signal (volts) [no restrictions]
c       val(i,20)= end vent lvdt signal (volts) [restricted]
c       val(i,21)= linear function actuator displacement (in)
c       val(i,22)= linear function end vent open area ratio
c       val(i,23)= linear function rwe open area ratio (end and side vents)
c       val(i,24)= linear function end vent lvdt signal (volts)
c
```

Appendix F: RWEAREA Program Listing

```
      Program RWEAREA
C
C    This program calculates the correct area ratio between the
C    expansion section of a shock tube or blast simulator and the open
C    area of a rarefaction wave eliminator located at its
C    downstream end. In the calculations the RWE is assumed to be
C    a simple converging nozzle open to the atmosphere at the
C    downstream end. The flow behind the shock is assumed to be a
C    one-dimensional, steady, isentropic flow of a perfect gas.
C
C    This Program Creates an Output File **(rweout)** which can be
C     used for Printing the results.
C
      REAL MT,M1
      CHARACTER*10 QUEST
      CHARACTER*10 QUEST2
      OPEN(1,file='rweout',status='old')
      Print*, "Enter the Ratio of Specific Heats (GAMMA)"
      Read*,GAMMA
   10 Print*, "Enter the units used for pressure; PSI or",
     +" kPa or atmos"
      Read*, QUEST
      Print*, "Enter the Ambient Pressure (PO)"
      Read*,PO
      Print*," Enter the shock overpressure (PS)"
      Read*,PS
      IF (QUEST.EQ."PSI".OR.QUEST.EQ."psi") THEN
      PT = PO*(6.895E+3)
      P1=(6.895E+3)*(PS+PO)
      ELSE IF (QUEST.EQ."kPa".OR.QUEST.EQ."KPA".OR.QUEST.EQ."kpa") THEN
      PT = PO*(1.0E+3)
      P1=(PS+PO)*(1.0E+3)
      ELSE IF (QUEST.EQ."atmos".OR.QUEST.EQ."atmos.") THEN
      PT = 101.33E+3*PO
      P1=(101.33E+3)*(PS+PO)
      ELSE IF (QUEST.EQ."ATMOS".OR.QUEST.EQ."ATMOS.") THEN
      PT = (101.33E+3)*PO
      P1=(101.33E+3)*(PS+PO)
      ELSE
      PRINT*,"Error in the pressure units; Try again"
      GOTO 10
      ENDIF
      ALPHA = (GAMMA+1)/(GAMMA-1)
      BETA = (GAMMA-1)/(2.0*GAMMA)
      DELTA = (GAMMA/(GAMMA-1.0))
C
C    Calculation of the Mach number behind the shock (M1)
C
```

```
      P1T = P1/PT
      M1 = (P1T-1.0)/(GAMMA*(BETA*P1T*(ALPHA+P1T))**0.5)
C
C    Calculation of the stagnation Pressure behind the shock (PS1)
C
      PS1 = P1*((2.0+(GAMMA-1.0)*M1**2.0)/2.0)**DELTA
C
C    Calculation of the isentropic Mach number at the downsteam
C    end of the RWE (MT)
C
      PS1T = PS1/PT
      IF (PS1T.GE.1.892936) THEN
      MT = 1.0
      ELSE
      MT = ((2.0/(GAMMA-1.0))*(PS1T**(1/DELTA)-1.0))**0.5
      ENDIF
C
C    Calculation of RWE area ratio (RWEAR)
C
      RWEAR = (1.0/MT)*((2.0+(GAMMA-1.0)*MT**2.0)/2.0)**(ALPHA/2)*
     +(M1)*((2.0+(GAMMA-1.0)*M1**2.0)/2.0)**(-1.0*ALPHA/2)
C
C    Output Section
C
      WRITE (1,2)
    2 FORMAT(/"*    *    *    *    *    *    *    *    *    *    *    *    *    *"/)
      PRINT*, "Ratio of Specific Heats (GAMMA) = ",GAMMA
      WRITE (1,5) GAMMA
    5 FORMAT(/2X,"Ratio of Specific Heats (GAMMA) = ",F12.4)
      PRINT*, "Atmosphereic Pressure (PO) = ",PO,QUEST
      WRITE (1,15) PO,QUEST
   15 FORMAT(/2X,"Atmosphereic Pressure (PO) = ",F12.4,1X,A10)
      PRINT*, "Incident Shock Overpressure (PS) = ",PS,QUEST
      WRITE (1,25) PS,QUEST
   25 FORMAT(/2X,"Incident Shock Overpressure (PS) = ",F15.5,1X,A10)
      PRINT*, "Mach Number Behind the Incident Shock (M1) = ",M1
      WRITE (1,35) M1
   35 FORMAT(/2X," Mach Number Behind the Incident Shock (M1) = ",F12.4)
      IF (QUEST.EQ."PSI".OR.QUEST.EQ."psi") THEN
      PS1 = PS1/(6.985E+3)
      ELSE IF (QUEST.EQ."atmos".OR.QUEST.EQ."atmos.") THEN
      PS1 = PS1/101.33E+3
      ELSE IF (QUEST.EQ."ATMOS".OR.QUEST.EQ."ATMOS.") THEN
      PS1 = PS1/101.33E+3
      ELSE IF (QUEST.EQ."kPa".OR.QUEST.EQ."KPA".OR.QUEST.EQ."kpa") THEN
      PS1 = PS1/(1.0E+3)
      ENDIF
      Print*, "Stagnation Pressure Behind the Incident Shock (PS1) = ",
```

```
  +PS1,QUEST
   WRITE (1,45) PS1,QUEST
45 FORMAT(/2X,"Stagnation Pressure behind the Incident Shock (PS1) =",
  +F12.4,1X,A10)
   Print*, "Mach Number at the Throat (MT) = ",MT
   WRITE (1,55) MT
55 FORMAT(/2X,"Mach Number at the Throat (MT) = ",F12.4)
   Print*, "Ratio between the Expansion Section Cross Area and the"
   PRINT*,"RWE Open Area Necessary for Proper Operation"
   PRINT*," (RWEAR) = ",RWEAR
   WRITE (1,65) RWEAR
65 FORMAT(/2X,"Ratio between the Expansion Section Cross Area and"/
  +5x, "the RWE Open Area Necessary for Proper Operation"/
  +5X,"(RWEAR) = ",F12.4)
   WRITE (1,2)
   Print*, "Calculate RWE area ratio at another shock overpressue ?"
   Print*, "Answer yes or no"
   Read*, QUEST2
   IF (QUEST2.EQ."yes".OR.QUEST2.EQ."YES".OR.QUEST2.EQ."Yes") THEN
   GOTO 10
   ELSE
   CLOSE(unit=6)
   ENDIF
   STOP
   END
```

NO. OF COPIES	ORGANIZATION
2	ADMINISTRATOR DEFENSE TECHNICAL INFO CTR ATTN DTIC DDA CAMERON STATION ALEXANDRIA VA 22304-6145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TA 2800 POWDER MILL RD ADELPHI MD 20783-1145
3	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TL 2800 POWDER MILL RD ADELPHI MD 20783-1145
1	DIRECTOR US ARMY RESEARCH LAB ATTN AMSRL OP SD TP 2800 POWDER MILL RD ADELPHI MD 20783-1145
	ABERDEEN PROVING GROUND
5	DIR USARL ATTN AMSRL OP AP L (305)

NO. OF COPIES	ORGANIZATION
2	HQDA ATTN SARD TR MS K KOMINOS DR R CHAIT WASHINGTON DC 20310-0103
2	HQDA ATTN SARD TT MS C NASH DR F MILTON WASHINGTON DC 20310-0103
2	DIRECTOR FEDERAL EMERGENCY MNGMNT AGENCY ATTN PUBLIC RELATIONS OFFICE TECHNICAL LIBRARY WASHINGTON DC 20472
1	CHAIRMAN DOD EXPLOSIVES SAFETY BOARD ROOM 856 C HOFFMAN BLDG 1 2461 EISENHOWER AVENUE ALEXANDRIA VA 22331-0600
1	DIRECTOR OF DEFENSE RESEARCH AND ENGINEERING ATTN DD TWP WASHINGTON DC 20301
1	DIRECTOR DEFENSE INTELLIGENCE AGENCY ATTN DT 2 WPNS & SYS DIVISION WASHINGTON DC 20301
1	ASSISTANT SECRETARY OF DEFENSE ATOMIC ENERGY ATTN DOCUMENT CONTROL WASHINGTON DC 20301
9	DIRECTOR DEFENSE NUCLEAR AGENCY ATTN CSTI TECHNICAL LIBRARY DDIR DFSP NANS OPNA SPSD SPTD DFTD TDTR WASHINGTON DC 20305

NO. OF COPIES	ORGANIZATION
1	CHAIRMAN JOINT CHIEFS OF STAFF ATTN J5 R&D DIVISION WASHINGTON DC 20301
2	DA DCSOPS ATTN TECHNICAL LIBRARY DIR OF CHEM & NUC OPS WASHINGTON DC 20310
3	COMMANDER FIELD COMMAND DNA ATTN FCPR FCTMOF NMHE KIRTLAND AFB NM 87115
1	U S ARMY RESEARCH DEVELOPMENT AND STANDARDIZATION GROUP UK ATTN DR ROY E REICHENBACH PSC 802 BOX 15 FPO AE 09499-1500
10	CENTRAL INTELLIGENCE AGENCY DIR DB STANDARD ATTN GE 47 HQ WASHINGTON DC 20505
1	DIRECTOR ADVANCED RESEARCH PROJECTS AGENCY ATTN TECHNICAL LIBRARY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714
2	COMMANDER US ARMY NRDEC ATTN AMSNA D DR D SIELING STRNC UE J CALLIGEROS NATICK MA 01762
2	COMMANDER US ARMY CECOM ATTN AMSEL RD AMSEL RO TPPO P FT MONMOUTH NJ 07703-5301
1	COMMANDER US ARMY CECOM R&D TECHNICAL LIBRARY ATTN ASQNC ELC IS L R MYER CTR FT MONMOUTH NJ 07703-5000

NO. OF COPIES	ORGANIZATION
1	MIT ATTN TECHNICAL LIBRARY CAMBRIDGE MA 02139
1	COMMANDER US ARMY NGIC ATTN RESEARCH & DATA BRANCH 220 7TH STREET NE CHARLOTTESVILLE VA 22901-5396
1	COMMANDER US ARMY ARDEC ATTN SMCAR FSM W BARBER BLDG 94 PICATINNY ARSENAL NJ 07806-5000
1	DIRECTOR US ARMY TRAC FT LEE ATTN ATRC L MR CAMERON FT LEE VA 23801-6140
1	US ARMY MISSILE & SPACE INTELLIGENCE CENTER ATTN AIAMS YDL REDSTONE ARSENAL AL 35898-5500
1	COMMANDING OFFICER CODE L51 NAVAL CIVIL ENGINEERING LABORATORY ATTN J TANCRETO PORT HUENEME CA 93043-5003
2	COMMANDER US ARMY STRATEGIC DEFENSE COMMAND ATTN CSSD H MPL TECH LIB CSSD H XM DR DAVIES PO BOX 1500 HUNTSVILLE AL 35807
3	COMMANDER US ARMY CORPS OF ENGINEERS WATERWAYS EXPERIMENT STATION ATTN CEWES SS R J WATT CEWES SE R J INGRAM CEWES TL TECH LIBRARY PO BOX 631 VICKSBURG MS 39180-0631
1	COMMANDER US ARMY ENGINEER DIVISION ATTN HNDED FD PO BOX 1500 HUNTSVILLE AL 35807

NO. OF COPIES	ORGANIZATION
3	COMMANDER US ARMY NUCLEAR & CHEMICAL AGENCY 7150 HELLER LOOP SUITE 101 SPRINGFIELD VA 22150-3198
1	COMMANDER US ARMY CORPS OF ENGINEERS FT WORTH DISTRICT ATTN CESWF PM J PO BOX 17300 FT WORTH TEXAS 76102-0300
1	DIRECTOR TRAC FLVN ATTN ATRC FT LEAVENWORTH KS 66027-5200
1	COMMANDER US ARMY RESEARCH OFFICE ATTN SLCRO D PO BOX 12211 RESEARCH TRIANGLE PARK NC 27709-2211
1	COMMANDER NAVAL ELECTRONIC SYSTEMS COMMAND ATTN PME 117 21A WASHINGTON DC 20360
1	DIRECTOR HQ TRAC RPD ATTN ATRC RPR RADDA FT MONROE VA 23651-5143
2	OFFICE OF NAVAL RESEARCH ATTN DR A FAULSTICK CODE 23 800 N QUINCY STREET ARLINGTON VA 22217
1	DIRECTOR TRAC WSMR ATTN ATRC WC KIRBY WSMR NM 88002-5502
1	COMMANDER NAVAL SEA SYSTEMS COMMAND ATTN CODE SEA 62R DEPARTMENT OF THE NAVY WASHINGTON DC 20362-5101

NO. OF COPIES	ORGANIZATION
1	COMMANDER US ARMY WSMR ATTN STEWS NED DR MEASON WSMR NM 88002-5158
2	CHIEF OF NAVAL OPERATIONS DEPARTMENT OF THE NAVY ATTN OP 03EG OP 985F WASHINGTON DC 20350
1	COMMANDER DAVID TAYLOR RESEARCH CENTER ATTN CODE 522 TECH INFO CTR BETHESDA MD 20084-5000
1	OFFICER IN CHARGE CODE L31 CIVIL ENGINEERING LABORATORY NAVAL CONSTRUCTION BATTALION CTR ATTN TECHNICAL LIBRARY PORT HUENEME CA 93041
1	COMMANDING OFFICER WHITE OAK WARFARE CENTER ATTN CODE WA501 NNPO SILVER SPRING MD 20902-5000
1	COMMANDER CODE 533 NAVAL WEAPONS CENTER ATTN TECHNICAL LIBRARY CHINA LAKE CA 93555-6001
1	COMMANDER DAHLGREN DIVISION NAVAL SURFACE WARFARE CENTER ATTN CODE E23 LIBRARY DAHLGREN VA 22448-5000
1	COMMANDER NAVAL RESEARCH LABORATORY ATTN CODE 2027 TECHNICAL LIBRARY WASHINGTON DC 20375
1	OFFICER IN CHARGE WHITE OAK WARFARE CTR DETACHMENT ATTN CODE E232 TECHNICAL LIBRARY 10901 NEW HAMPSHIRE AVENUE SILVER SPRING MD 20903-5000
1	AL LSCF ATTN J LEVINE EDWARDS AFB CA 93523-5000

NO. OF COPIES	ORGANIZATION
1	COMMANDER NAVAL WEAPONS EVALUATION FAC ATTN DOCUMENT CONTROL KIRTLAND AFB NM 87117
1	RADC EMTLD DOCUMENT LIBRARY GRIFFISS AFB NY 13441
1	AEDC ARNOLD AFB TN 37389
1	AFESC RDCS ATTN PAUL ROSENGREN TYNDALL AFB FL 32403
1	OLAC PL TSTL ATTN D SHIPLETT EDWARDS AFB CA 93523-5000
1	AFIT ENY ATTN LTC HASEN PHD WRIGHT PATTERSON AFB OH 45433-6583
2	AIR FORCE ARMAMENT LABORATORY ATTN AFATL DOIL AFATL DLYV EGLIN AFB FL 32542-5000
1	DIRECTOR IDAHO NATIONAL ENGINEERING LAB ATTN SPEC PROGRAMS J PATTON 2151 NORTH BLVD MS 2802 IDAHO FALLS ID 83415
3	PHILLIPS LABORATORY AFWL ATTN NTE NTED NTES KIRTLAND AFB NM 87117-6008
1	DIRECTOR LAWRENCE LIVERMORE NATIONAL LAB ATTN TECH INFO DEPT L 3 PO BOX 808 LIVERMORE CA 94550
1	AFIT ATTN TECHNICAL LIBRARY BLDG 640 B WRIGHT PATTERSON AFB OH 45433

NO. OF COPIES	ORGANIZATION
1	DIRECTOR NATIONAL AERONAUTICS & SPACE ADMIN ATTN SCIENTIFIC & TECH INFO FAC PO BOX 8757 BWI AIRPORT BALTIMORE MD 21240
1	FTD NIIS WRIGHT PATTERSON AFB OH 45433
3	KAMAN SCIENCES CORPORATION ATTN LIBRARY PA ELLIS FH SHELTON PO BOX 7463 COLORADO SPRINGS CO 80933-7463
4	DIRECTOR IDAHO NATIONAL ENGINEERING LAB EG&G IDAHO INC ATTN R GUENZLER MS 3505 R HOLMAN MS 3510 R A BERRY W C REED PO BOX 1625 IDAHO FALLS ID 83415
5	DIRECTOR SANDIA NATIONAL LABS ATTN DOC CONTROL 3141 C CAMERON DIV 6215 A CHABAI DIV 7112 D GARDNER DIV 1421 J MCGLAUN DIV 1541 PO BOX 5800 ALBUQUERQUE NM 87185-5800
2	DIRECTOR LOS ALAMOS NATIONAL LABORATORY ATTN TH DOWLER MS F602 DOC CONTROL FOR REPORTS LIBRARY PO BOX 1663 LOS ALAMOS NM 87545
1	BLACK & VEATCH ENGINEERS ARCHITECTS ATTN HD LAVERENTZ 1500 MEADOW LAKE PARKWAY KANSAS CITY MO 64114

NO. OF COPIES	ORGANIZATION
1	DIRECTOR SANDIA NATIONAL LABORATORIES LIVERMORE LABORATORY ATTN DOC CONTROL FOR TECH LIB PO BOX 969 LIVERMORE CA 94550
1	DIRECTOR NASA AMES RESEARCH CENTER APPLIED COMPUTATIONAL AERO BRANCH ATTN DR T HOLTZ MS 202 14 MOFFETT FIELD CA 94035
1	DIRECTOR NASA LANGLEY RESEARCH CENTER ATTN TECHNICAL LIBRARY HAMPTON VA 23665
2	APPLIED RESEARCH ASSOCIATES INC ATTN J KEEFER NH ETHRIDGE PO BOX 548 ABERDEEN MD 21001
1	ADA TECHNOLOGIES INC ATTN JAMES R BUTZ HONEYWELL CENTER SUITE 110 304 INVERNESS WAY SOUTH ENGLEWOOD CO 80112
1	ALLIANT TECHSYSTEMS INC ATTN ROGER A RAUSCH MN48 3700 7225 NORTHLAND DRIVE BROOKLYN PARK MN 55428
1	CARPENTER RESEARCH CORPORATION ATTN H JERRY CARPENTER 27520 HAWTHORNE BLVD SUITE 263 PO BOX 2490 ROLLING HILLS ESTATES CA 90274
1	AEROSPACE CORPORATION ATTN TECH INFO SERVICES PO BOX 92957 LOS ANGELES CA 90009
1	GOODYEAR CORPORATION ATTN RM BROWN BLDG 1 SHELTER ENGINEERING LITCHFIELD PARK AZ 85340

NO. OF COPIES	ORGANIZATION
1	THE BOEING COMPANY ATTN AEROSPACE LIBRARY PO BOX 3707 SEATTLE WA 98124
2	FMC CORPORATION ADVANCED SYSTEMS CENTER ATTN J DROTLEFF C KREBS MDP 95 BOX 58123 2890 DE LA CRUZ BLVD SANTA CLARA CA 95052
1	CALIFORNIA RES & TECH INC ATTN M ROSENBLATT 20943 DEVONSHIRE STREET CHATSWORTH CA 91311
1	SVERDRUP TECHNOLOGY INC SVERDRUP CORPORATION AEDC ATTN BD HEIKKINEN MS 900 ARNOLD AFB TN 37389-9998
1	DYNAMICS TECHNOLOGY INC ATTN D T HOVE G P MASON 21311 HAWTHORNE BLVD SUITE 300 TORRANCE CA 90503
1	KTECH CORPORATION ATTN DR E GAFFNEY 901 PENNSYLVANIA AVE NE ALBUQUERQUE NM 87111
1	EATON CORPORATION DEFENSE VALVE & ACTUATOR DIV ATTN J WADA 2338 ALASKA AVE EL SEGUNDO CA 90245-4896
2	MCDONNELL DOUGLAS ASTRONAUTICS CORP ATTN ROBERT W HALPRIN KA HEINLY 5301 BOLSA AVENUE HUNTINGTON BEACH CA 92647

NO. OF COPIES	ORGANIZATION
4	KAMAN AVIDYNE ATTN R RUETENIK 2 CP S CRISCIONE R MILLIGAN 83 SECOND AVENUE NORTHWEST INDUSTRIAL PARK BURLINGTON MA 01830
1	MDA ENGINEERING INC ATTN DR DALE ANDERSON 500 EAST BORDER STREET SUITE 401 ARLINGTON TX 07601
2	PHYSICS INTERNATIONAL CORPORATION PO BOX 5010 SAN LEANDRO CA 94577-0599
2	KAMAN SCIENCES CORPORATION ATTN DASIAC 2 CP PO DRAWER 1479 816 STATE STREET SANTA BARBARA CA 93102-1479
1	R&D ASSOCIATES ATTN GP GANONG PO BOX 9377 ALBUQUERQUE NM 87119
1	LOCKHEED MISSILES & SPACE CO ATTN J J MURPHY DEPT 81 11 BLDG 154 PO BOX 504 SUNNYVALE CA 94086
2	SCIENCE CENTER ROCKWELL INTERNATIONAL CORP ATTN DR S CHAKRAVARTHY DR D OTA 1049 CAMINO DOS RIOS THOUSAND OAKS CA 91358
1	ORLANDO TECHNOLOGY INC ATTN D MATUSKA 60 SECOND STREET BLDG 5 SHALIMAR FL 32579

NO. OF COPIES	ORGANIZATION
3	S CUBED A DIVISION OF MAXWELL LABS INC ATTN TECHNICAL LIBRARY R DUFF K PYATT PO BOX 1620 LA JOLLA CA 92037-1620
2	THE RALPH M PARSONS COMPANY ATTN T M JACKSON LB TS PROJECT MANAGER 100 WEST WALNUT STREET PASADENA CA 91124
1	SAIC ATTN J GUEST 2301 YALE BLVD SE SUITE E ALBUQUERQUE NM 87106
1	SUNBURST RECOVERY INC ATTN DR C YOUNG PO BOX 2129 STEAMBOAT SPRINGS CO 80477
1	SAIC ATTN N SINHA 501 OFFICE CENTER DRIVE APT 420 FT WASHINGTON PA 19034-3211
1	SVERDRUP TECHNOLOGY INC ATTN RF STARR PO BOX 884 TULLAHOMA TN 37388
2	S CUBED A DIVISION OF MAXWELL LABS INC ATTN C E NEEDHAM L KENNEDY 2501 YALE BLVD SE ALBUQUERQUE NM 87106
3	SRI INTERNATIONAL ATTN DR GR ABRAHAMSON DR J GRAN DR B HOLMES 333 RAVENWOOD AVENUE MENLO PARK CA 94025

NO. OF COPIES	ORGANIZATION
1	TRW BALLISTIC MISSILE DIVISION ATTN H KORMAN MAIL STATION 526 614 PO BOX 1310 SAN BERNADINO CA 92402
1	BATTELLE TWSTIAC 505 KING AVENUE COLUMBUS OH 43201-2693
1	THERMAL SCIENCE INC ATTN R FELDMAN 2200 CASSENS DRIVE ST LOUIS MO 63026
2	DENVER RESEARCH INSTITUTE ATTN J WISOTSKI TECHNICAL LIBRARY PO BOX 10758 DENVER CO 80210
1	STATE UNIVERSITY OF NEW YORK MECHANICAL & AEROSPACE ENGINEERING ATTN DR PEYMAN GIVI BUFFALO NY 14260
2	UNIVERSITY OF MARYLAND INSTITUTE FOR ADV COMPUTER STUDIES ATTN L DAVIS G SOBIESKI COLLEGE PARK MD 20742
2	THINKING MACHINES CORPORATION ATTN G SABOT R FERREL 245 FIRST STREET CAMBRIDGE MA 02142-1264
1	NORTHROP UNIVERSITY ATTN DR FB SAFFORD 5800 W ARBOR VITAE STREET LOS ANGELES CA 90045
1	CALIFORNIA INSTITUTE OF TECHNOLOGY ATTN T J AHRENS 1201 E CALIFORNIA BLVD PASADENA CA 91109

NO. OF COPIES	ORGANIZATION
1	STANFORD UNIVERSITY ATTN DR D BERSHADER DURAND LABORATORY STANFORD CA 94305
1	UNIVERSITY OF MINNESOTA ARMY HIGH PERF COMP RES CTR ATTN DR TAYFUN E TEZDUYAR 1100 WASHINGTON AVE SOUTH MINNEAPOLIS MN 55415
3	SOUTHWEST RESEARCH INSTITUTE ATTN DR C ANDERSON S MULLIN A B WENZEL PO DRAWER 28255 SAN ANTONIO TX 78228-0255
2	COMMANDER US ARMY NRDEC ATTN SSCNC YSD J ROACH SSCNC WST A MURPHY KANSAS STREET NATICK MA 10760-5018

NO. OF COPIES	ORGANIZATION
	ABERDEEN PROVING GROUND
1	CDR USATECOM ATTN AMSTE TE F L TELETSKI
1	CDR USATHAMA ATTN AMSTH TE
1	CDR USATC ATTN STEC LI
26	DIR USARL ATTN AMSRL SC C H BREAUX AMSRL SC CC C NIETUBICZ C ELLIS D HISLEY N PATEL T KENDALL R SHEROKE AMSRL SC I W STUREK AMSRL SC AE M COLEMAN AMSRL SC S R PEARSON AMSRL SL CM E FIORVANTE AMSRL WT N J INGRAM AMSRL WT NA R KEHS AMSRL WT NC R LOTTERO B MCGUIRE A MIHALCIN P MULLER R LOUCKS S SCHRAML AMSRL WT ND J MILETTA AMSRL WT NF L JASPER AMSRL WT NG T OLDHAM AMSRL WT NH J CORRIGAN AMSRL WT PB P WEIHNACHT B GUIDOS AMSRL WT TC K KIMSEY

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. ARL Report Number ARL-MR-260 Date of Report September 1995

2. Date Report Received ____________________

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.) ____________________

4. Specifically, how is the report being used? (Information source, design data, procedure, source of ideas, etc.) ____________________

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided, or efficiencies achieved, etc? If so, please elaborate. ____________________

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.) ____________________

CURRENT ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

7. If indicating a Change of Address or Address Correction, please provide the Current or Correct address above and the Old or Incorrect address below.

OLD ADDRESS

Organization

Name

Street or P.O. Box No.

City, State, Zip Code

(Remove this sheet, fold as indicated, tape closed, and mail.)
(DO NOT STAPLE)